The Resource Journal For IBM PC Programmers

PROGRAMMER'S Journal

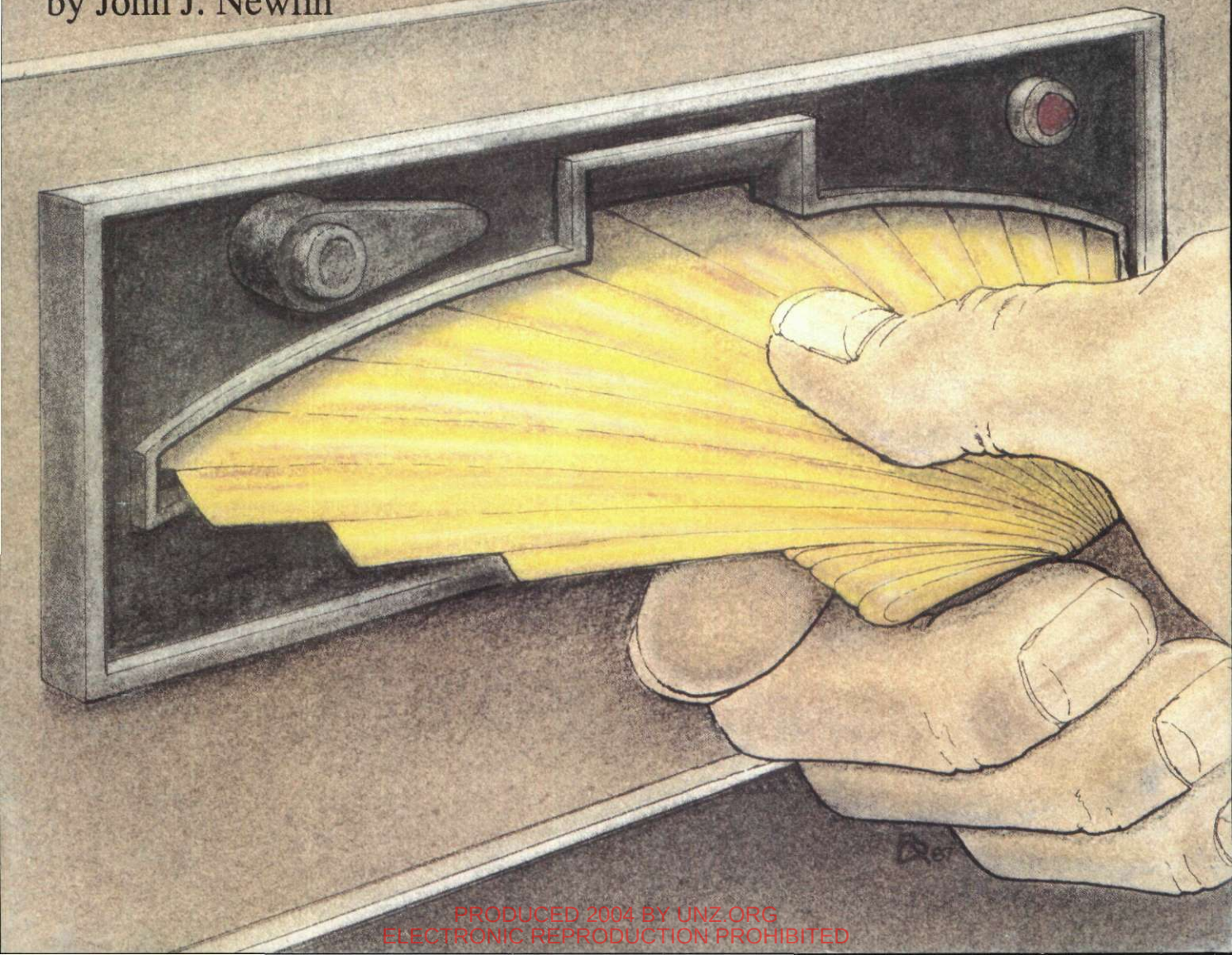
NOVEMBER/DECEMBER 1987
VOLUME 5.6
$3.95

The Shell Game
by John J. Newlin

The Second Coming
by Christian Walker

TurboPascal 4.0
by Tom Swan

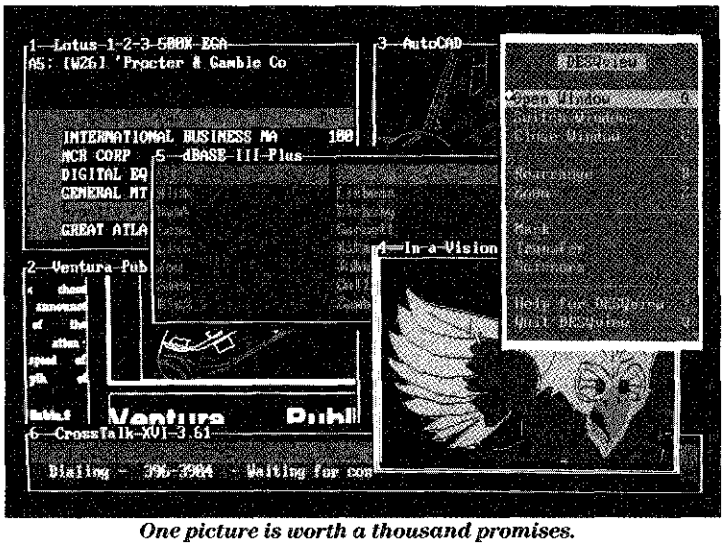
Usenix and Pink Noise
by Frank D. Greco



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

How to tell the difference between DESQview™ 2.0 and any other environment.

Selecting DESQview, the environment of choice, can give you the productivity and power you crave, without the loss of your old programs and hardware. If you like your existing programs, want to use them together, transfer data between them, print, sort, communicate with or process-in-background, yet still have the need to keep in place your favorite PC(8088, 8086, 80286 or 80386), DESQview is the "proven true" multitasking, multi-windowing environment for you. Best of all, DESQview 2.0 is here now, with all the money saving, time saving, and productivity features that others can only promise for the all-too-distant future.



One picture is worth a thousand promises.

And with DESQview's new graphics enhancements for Hercules, CGA, EGA, and VGA, Version 2.0 still offers the same award winning and pioneering features for programs that earned DESQview its leadership, only now you can also run desktop publishing programs, CAD programs, even GEM-™, Topview-™, and Microsoft Windows-™ specific programs. In some cases you'll add as little as 10–40K to your system overhead. Now you can have multi-tasking, multi-windowing, break the 640K habit too and still get an auto dialer, macros, menus for DOS and, for advanced users, a new complete application programmer's interface capability. No wonder that over the years, and especially in recent months, DESQview, and now DESQview 2.0 have earned extravagant praise from some of the most respected magazines in the industry.

"Product of the Year" by readers vote in InfoWorld.

"Best PC Environment" by popular vote at Comdex Fall in PC Tech Journal's "System Builder" Contest.

"—I wouldn't want to run an IBM or compatible computer without DESQview"—InfoWorld, Michael Miller.

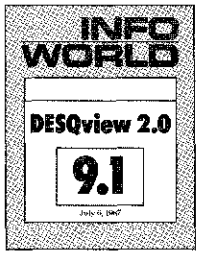
"A colossus among windowing environments"... "will run almost anything"—PC Week, Marvin Bryan.

"Windows, promises, but DESQview delivers"—MICRO-TIMES, Birrell Walsh.

No other environment has consistently pioneered features, openness, and productivity. See for yourself. Send in the coupon. The possibilities are endless with DESQview 2.0.

Attention Programmers: For more information about Quarterdeck's API, and future 386 program extensions, call us today.

SYSTEM REQUIREMENTS
IBM Personal Computer and 100% compatibles (with 8086, 8088, 80286 or 80386 processors) with monochrome or color display; IBM Personal System/2 • Memory: 640K recommended; for DESQview itself 0-145K • Expanded Memory (Optional): expanded memory boards compatible with the Intel AboveBoard; enhanced expanded memory boards compatible with the AST RAMpage • Disk: Two diskette drives or one diskette drive and a hard disk • Graphics Card (Optional): Hercules, IBM Color/Graphics (CGA), IBM Enhanced Graphics (EGA), IBM Personal System/2 Advanced Graphics (VGA) • Mouse (Optional): Mouse Systems, Microsoft and compatibles • Modem for Auto-Dialer (Optional): Hayes or Compatible • Operating System: PC-DOS 2.0-3.3; MS-DOS2.0-3.2 • Software: Most PC-DOS and MS-DOS application programs; programs specific to TopView 1.1, GEM 1.1 and Microsoft Windows 1.03 • Media: DESQview 2.0 is available on either 5¼" or 3½" floppy diskettes



Rush me DESQview 2.0! Today! PGMR - 11/12

No. of Copies	Media 3½"/5¼"	Product	Retail Price ea.	Total
		DESQview 2.0	$129.95	$
		Shipping & Handling USA	$ 5.00	$
		Outside USA	$ 10.00	
		Sales Tax (CA residents)	6.5%	$
			Amount Enclosed	$

Payment: ☐ Visa ☐ MC ☐ AMEX ☐ Check

Credit Card: Valid Since ____/____ Expiration ____/____

Card Number:

Credit Card Name

Shipping Address

City State Zip Telephone

Mail to: Quarterdeck Office Systems, 150 Pico Boulevard, Santa Monica, CA 90405.
NOTE: If you own DESQview call us for a special upgrade offer, or send in your DESQview registration card. AST Special Edition users included.

DESQ™ view
Quarterdeck

Quarterdeck Office Systems • 150 Pico Boulevard, Santa Monica, CA 90405 • (213) 392-9851

DESQview is a trademark of Quarterdeck Office Systems. AboveBoard is a trademark of Intel Corporation. Hayes is a trademark of Hayes MicroComputer Products Inc. IBM, PC, Personal System/2 and TopView are trademarks of International Business Machines Corporation. Microsoft Windows and MS are registered trademarks of Microsoft Corporation. Mouse Systems is a trademark of Metagraphics/Mouse Systems. RAMpage is a trademark of AST Research, Inc. GEM is a trademark of Digital Research. Hercules is a trademark of Hercules.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

DEBUGGING SWAT TEAM

Order Eco-C88 Rel. 4.0 New Modeling Compiler and get C-more at no extra charge!

Seek and Correct

You already know that fast compilation does not mean fast program development. Backing up for bogus error messages and removing the bugs takes time. Eco-C88's "Seek and Correct" three - way error checking finds even the most elusive bugs, clearing the path for swift program development.

Double Barrel Error Checking

Eco-C88 nails **syntax errors** cold and tells you about the error in plain English. And there's no avalanche of false error messages, either. Other compilers can generate up to four times the number of error messages actually present; they leave it up to you to guess which ones are real. You'll be more productive with Eco-C88 because there is no guess work.

Eco-C88 provides ten levels of **semantic error** checking. You can select from almost no checking to the fussiest you've ever seen. Eco-C88's "picky flag" finds subtle errors that slip by other compilers.

Eco-C88 also features:

- All data types, plus ANSI Enhancements
- Robust library, including many new ANSI functions
- CED editor with online function help, split windows, compile-edit-link capability
- New, expanded manual with sample programs for the library functions

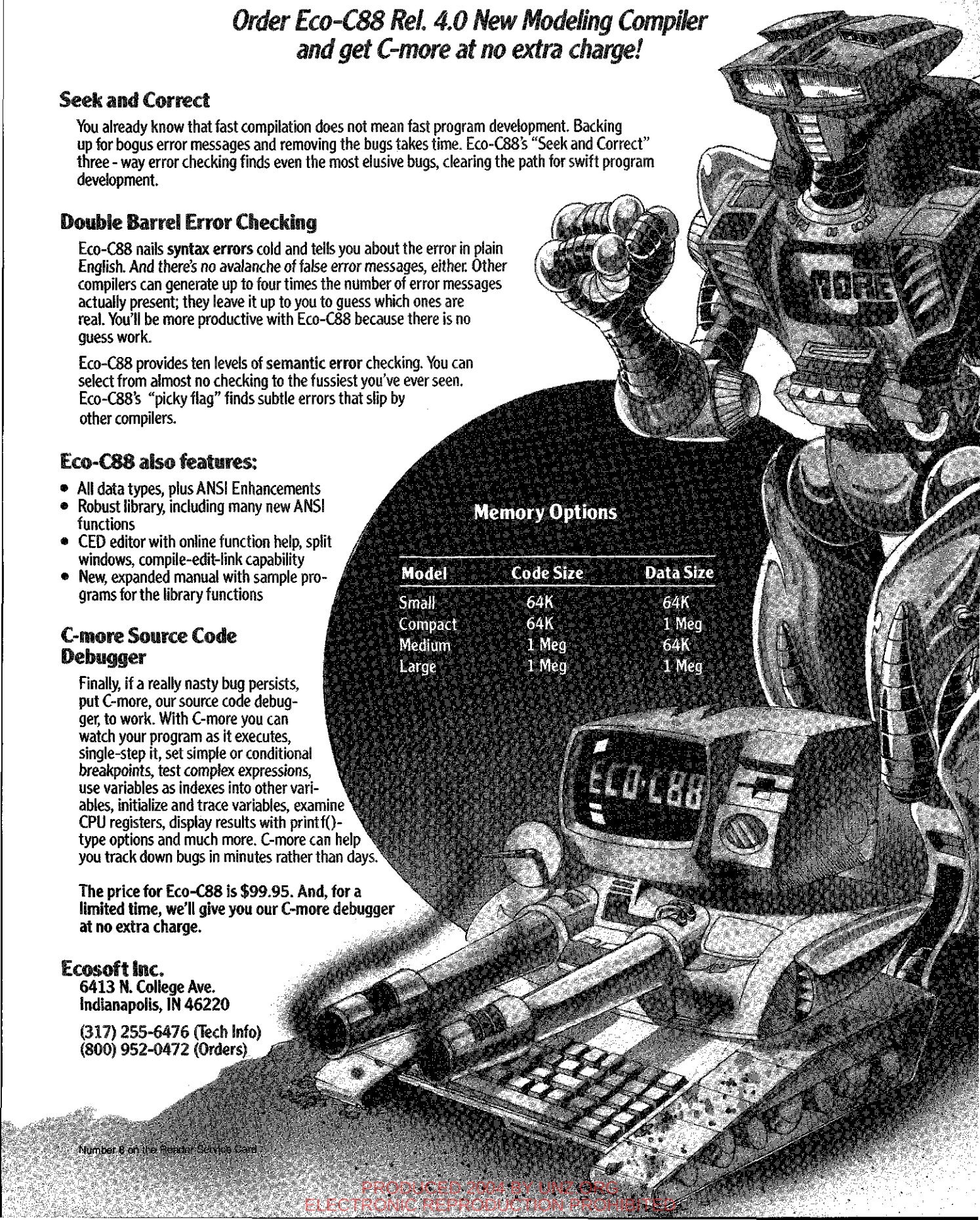
C-more Source Code Debugger

Finally, if a really nasty bug persists, put C-more, our source code debugger, to work. With C-more you can watch your program as it executes, single-step it, set simple or conditional breakpoints, test complex expressions, use variables as indexes into other variables, initialize and trace variables, examine CPU registers, display results with printf()-type options and much more. C-more can help you track down bugs in minutes rather than days.

The price for Eco-C88 is $99.95. And, for a limited time, we'll give you our C-more debugger at no extra charge.

Memory Options

Model	Code Size	Data Size
Small	64K	64K
Compact	64K	1 Meg
Medium	1 Meg	64K
Large	1 Meg	1 Meg



Ecosoft Inc.
6413 N. College Ave.
Indianapolis, IN 46220

(317) 255-6476 (Tech Info)
(800) 952-0472 (Orders)

Number 6 on the Reader Service Card.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The Resource Journal For IBM PC Programmers

PROGRAMMER'S Journal

Volume 5.6
November/December 1987

Contents

4 **Note from the Editors**

6 **Letters**

10 **PJ Forum:** The Second Coming—Multics, OS/2, Paging and Segmentation *by Christian Walker*
Surely OS/2 is at hand . . .

14 **Redirections:** Report from the Usenix Conference and Pink Noise *by Frank D. Greco*
Show notes and other news from our Unix correspondent

20 **Consulting:** How to Get Rich in Consulting *by Paul Barkley*
Sensible steps and one draconian solution

26 **Pascal:** The Shell Game *by John J. Newlin*
MS-DOS, meet Marshal Pascal

54 **Pascal:** Turbo Pascal's Renaissance *by Tom Swan*
Tom takes us back to the future

60 **C:** Handling Fixed Length Character Arrays *by William J. Hunt*
The **memxxx** library functions

62 **C:** The Proposed ANSI C Language Standard *by Rex Jaeschke*
Rex takes aim at a moving target

66 **Miscellaneous:** *by M. Steven Baker*

77 **Reader Survey**

79 **PJ Billboard**

80 **Advertiser Index**

DOS shells are handy items. Your high-level language program will enjoy a chance to talk to the big world. Artist Ralph Bentley presents his vision of John Newlin's article, "The Shell Game."

Page 26

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Blaise puts the Accent on C with C TOOLS PLUS/5.0™

Now, just $129 and supports Microsoft C 5.0 and QuickC

Enhance your Microsoft C programming environment with C TOOLS PLUS/5.0™—a new, quintessential library of C functions. C TOOLS PLUS/5.0 from Blaise Computing Inc. puts a prime accent on quickly building professional applications using the full power of Microsoft C Version 5.0 and QuickC. Now you can concentrate on program creativity by having full control over DOS, menus, interrupt service routines, memory resident programs, printer and keyboard control, and more!

C TOOLS PLUS/5.0 prebuilt libraries are ready to use with either QuickC or the Microsoft C Version 5.0 command line environment. Complete documented source code is included so that you can study and adapt it to your specific needs. Blaise Computing's attention to detail, like the use of full function prototyping, cleanly organized header files, and a comprehensive, fully-indexed manual, makes C TOOLS PLUS/5.0 the choice for experienced developers as well as newcomers to C.

Continuous refinement of Blaise Computing's library products has produced a collection of tools that are unsurpassed for reliability, functionality and ease of use. Built upon the widely acclaimed C TOOLS PLUS, C TOOLS PLUS/5.0 includes such highly-developed features as:

- Windowing routines, cursor memory.
- Accept user input, formatted output.
- "printf" window-oriented output. **NEW!**

◆INTERRUPT SERVICE ROUTINES
- Capture DOS critical errors and keystrokes.
- Install hardware interrupt handlers.

◆RESIDENT SOFTWARE SUPPORT
- Install, detect and remove memory resident programs.

◆MENUS
- Horizontal and pulldown. **NEW!**
- Lotus-style support. **NEW!**

◆INTERVENTION CODE
- Schedule C functions at specified times, intervals or with a "hot key." **NEW!**
- Take full advantage of DOS, even from memory resident programs. **NEW!**

◆FAST DIRECT VIDEO ACCESS
- All monitors, even EGA 43-line mode.

◆PRINTER CONTROL
- Access BIOS print functions. **NEW!**
- Control the DOS PRINT utility. **NEW!**

◆UTILITIES AND MACROS
- Take advantage of DOS file structure.
- Manipulate data types, far & near pointers. **NEW!**
- Access any memory areas with fast "peek" and "poke" macros. **NEW!**

C TOOLS PLUS/5.0 supports the Microsoft C Version 5.0 and QuickC compilers, requires DOS 2.00 or later and is just **$129.00.**

C ASYNCH MANAGER™ Version 2.0 IMPROVED!

C ASYNCH MANAGER is a library of functions designed to help you incorporate asynchronous communication capabilities into your application programs. Version 2.0 has been rewritten especially for Microsoft C Version 5.0 and Borland's Turbo C. Simultaneous buffered input and output to both COM ports at speeds up to 9600 baud, XON/XOFF protocol, modem control and XMODEM file transfer are among the many features supported and is priced at just **$175.00.**

Blaise computing Inc. has a full line of support products for both Pascal and C. Call today for your free information packet.

BLAISE COMPUTING INC.

2560 Ninth Street, Suite 316 Berkeley, CA 94710 (415) 540-5441

Number **2** on the Reader Service Card.

BLAISE N U

$275.00

...screens, menus, pulldown menus, block mode data entry, field-by-field control with ... DOS, EGA 43-line, PS/2, or MS Pascal.

...$129.00

Windows, TSRs, interrupt code, screen handling and EGA 43-line text mode support, direct screen access, DOS file handling and more. For Turbo C.

Turbo POWER TOOLS

COMING SOON! General screen management, panel screens, block mode data entry or field-by-field control with instant screen access. For Turbo Pascal.

Turbo POWER TOOLS PLUS $99.95

Screen and window management including EGA support, DOS memory control, TSRs, scheduled intervention code and much more. For Turbo Pascal.

Turbo ASYNCH PLUS $99.95

Interrupt driven support for the COM ports, I/O buffers up to 64K, XON/XOFF, up to 9600 baud, modem and XMODEM control. For Turbo Pascal.

PASCAL TOOLS, TOOLS 2 $175.00

Expanded string and screen handling, graphics routines, memory management, general program control, DOS file support and more. For MS-Pascal.

ASYNCH MANAGER $175.00

Full featured interrupt driven support for the COM ports, I/O buffers up to 64K, XON/XOFF, up to 9600 baud, modem control and XMODEM. For MS-Pascal.

...$49.95

"Super-batch" program. Create batch files which can invoke programs and provide input to them, run any program unattended, create demonstration programs, analyze keyboard usage.

EXEC ...$95.00

NEW VERSION! Programs chaining executive. Chain one program from another in different languages, specify common data areas, less than 2K of overhead.

SCRIPT ...$49.95

Text formatter for all programmers. Flexible printer control, user-defined variables, index generation, general macro facility. Crafted in Turbo Pascal.

LIGHT, FAST & S... $99.95

Windows, TSRs, EGA 43-line text mode, direct screen access, DOS file handling and more. For the Datalight C compiler.

TECHNICAL TOLL FREE

(800) 333-4160

TELEX NUMBER 338139

Yes! Send me the prime accent!

Enclosed is $_____ for _____ copies of _____.

☐ Please send me more information on your products.

CA residents add Sales Tax. Domestic orders add $4.00 for UPS shipping, $10.00 for Federal Express standard air.

Name: _____ Phone: (_____) _____

Address: _____

City: _____ State: _____ Zip: _____

VISA or MC#: _____ Exp. Date: _____

Microsoft is a registered trademark of Microsoft Corporation. QuickC is a trademark of Microsoft Corporation. Turbo C is a registered trademark of Borland International.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The Resource Journal For IBM PC Programmers
PROGRAMMER'S Journal

Editors
Charles Aylworth, Erin O'Connor
Technical Editor
M. Steven Baker
Contributing Editors
Frank D. Greco, Tom Swan

Contributing Writers
Paul Barkley
Frank D. Greco
Rex Jaeschke
John J. Newlin
Tom Swan
Christian Walker

Publisher
Elizabeth Oakley
Production
Ralph Bentley
Illustration
Jim Carpenter
Advertising
Robert Tappan
Subscriptions
Gary Hoyer
Accounting
Bobbi Sinyard
Founder
Greg Estes

Business and Editorial Offices
Programmer's Journal
150 North 4th Street
Springfield, OR 97477
(503) 747-0800

Subscriptions: 1 year (6 issues) $24
($30 in Canada and $40 elsewhere)
Change of Address
Programmer's Journal
P.O. Box 30160
Eugene, OR 97403

Articles and letters in Programmer's Journal represent the views of their authors and not necessarily those of the publisher. Letters to the editor become the property of Programmer's Journal and may be edited and published.

Code and listings in Programmer's Journal appear "as is," and Programmer's Journal makes no guarantee or warranty of their quality, performance, accuracy, or suitability for any particular purpose. Programmer's Journal assumes no responsibility or liability to any person or entity with respect to any damage caused or alleged to be caused by its use.

Advertising: For ad rates, deadlines, and placement, call Robert Tappan, Ralph Bentley, or Gary Hoyer (503/747-0800), or write to Programmer's Journal, P.O. Box 30160, Eugene, OR 97403.

Programmer's Journal (ISSN 0747-586) is published bi-monthly by Oakley Publishing Company. Second-class postage paid at Eugene, OR, and additional mailing offices.

POSTMASTER: Send address changes to Programmer's Journal, P.O. Box 30160, Eugene, OR 97403.

© 1987 OAKLEY PUBLISHING COMPANY

Note from the Editors

One of our goals at *PJ* is to make the magazine easy to use as well as useful. An often considered problem has been how to make the extensive code listings we print accessible. We all know it's a chore to type in a ten page listing, and the chance of a typo approaches unity. Although authors generally enjoy feedback from readers, they don't care much for spending half an hour on the phone locating an O typed instead of a 0.

The first thought that springs to mind is to set up a SIG (or Forum, or what-have-you) on one of the commercial on-line services. It turns out that that is not as easy as it may seem. CompuServe and the Source require that you submit a proposal for the area, and they take considerable time to respond to that proposal. We haven't been able to find out what you do with BIX. The only nationwide service we have gotten through to is Mnematics. They have a *PJ* area, but at the moment, access to their service is a little confusing. (If you are already a Mnematics user, keep an eye on the area—we plan to get it updated just as soon as we figure out how to log on.) So much for national nets. We haven't given up, but we haven't made much progress either.

We have had some success with the bulletin board idea, though. First, we have code files posted on Turbo Source Search, a commercial board out of Scituate MA. The problem here is that the board is in Massachusetts, and most of our readers aren't. PC Pursuit notwithstanding, a long-distance call to MA is not quite reader-friendly. (Gerry Power, the Sysop of TSS, has been very helpful and deserves an "attaboy" at least. Call TSS at (617) 545-9131.

An idea that seems to make sense is to use the FIDO/OPUS network. This may turn out to be the solution. For those of you unfamiliar with modemland, the network is made up of many privately operated bulletin boards running a common program. When everything is working right, each board calls one or more other boards each night and exchanges messages. These "hub" boards call other hubs, and in theory the messages spread out across the land. *PJ* now has an area established on the local/national FIDO network. Our originating board is the Blazer in Eugene, OR (503/343-9724). It is part of Net 152. Each night any traffic in the *PJ* area is sent to a board in Tacoma (Hub 193) and re-sent to all boards connecting with that hub. The whole process is called "echomail." Only the hubs have to make more than one long-distance call.

We think that the *PJ* echomail may be just what we need, but the plan doesn't work unless it's worth the phone calls back and forth. Here's where you come in. You need to locate a local FIDO/OPUS board and let the Sysop know you want him to carry the *PJ* echo. If he agrees, he should be able to establish a minimum link connection to Hub 193. A few nights later, you can post "Howdy, PJ" and be blessed with your very own mumble from the editors. Each issue, we will post an ARC'ed library of the code files, just as we received them from the authors.

We have had several requests for the code to Michael Abrash's EGA articles. Michael has given his permission, so we are posting the code to our various areas and making this offer—for $10 to cover shipping and handling, we'll send you a disk of the code. Prepaid only.

Finally, *PJ* is now available on microform from University Microfilms International, 300 Zeeb Rd., Ann Arbor, MI 48106. Volumes available depend in part on the number of requests that have been made.

Let us know how these various access methods work out for you. We'll try any good idea that doesn't take a lot of bucks or a lot of time.—*Chuck*

It's Reader Survey time again, and we want to know everything. If you're reticent, skip the overcurious questions, but please fill out as much of the survey at the end of this issue as you feel you can in all decency. We're prepared to pay the postage and close attention to what you say.

The most frequent complaint on our Peer Review cards is that *PJ* isn't big enough, so we're giving you a blockbuster this time—and every issue from now on, accountant permitting. Our new size gives John Newlin plenty of room in which to introduce us to Marshal Pascal, and we're previewing a new version of an old favorite. Yes, it's Borland again. Tom Swan's a Beta tester, and we have the good fortune to be privy to his reactions as he puts Turbo Pascal 4.0 through its paces (assisted by the Macintosh Turbo Pascal manual!). We'll follow up with a review proper.

Turbo Pascal 4.0 may be fast enough to satisfy even Jon Greenblatt. Some of you were mystified or just plain appalled by Jon's "Note on Optimizing Turbo Pascal" (3.0) in Volume 5.5 (see LETTERS in this issue). Others of you (even purists) confessed to a sneaking admiration for Jon's brassy ingenuity. The faint of heart should be advised—next issue we're going to run Jon's "Hell-for-Leather Turbo."

Back at the Second Coming, Christian Walker continues a lengthening preamble (we're still waiting for an operating system for the 386) to his consideration of the respective merits of OS/2 and developing for the 386 with an optimal operating system for that machine. Steve Baker is keeping an eye on emerging operating systems and reports on DRI's Concurrent DOS for the 386 in his MISCELLANEOUS column this issue. Frank Greco covers late (or never) blooming operating systems in his REDIRECTIONS column.—*Erin*

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The Software Family

Dept PJ117, 649 Mission Street
San Francisco, CA 94105

(1)800-443-7176

In California or outside U.S.

(415)583-4166

Trust TSF to provide the best value and service!

* Technical advice and support by programmers
* Honest and equitable business practices
* Prompt, flexible service to meet your needs

We accept checks, Visa, MasterCard, American Express and COD. We charge your card only when we ship and do **not** add a surcharge. We provide free UPS delivery on software orders over $100 ($3 delivery charge on orders under $100). We add actual charges for UPS Blue Label or Federal Express rush service. Our COD fee is $2. We allow return privileges on most products. We carry a large inventory and ship promptly, so you receive your shipment within 3 to 6 working days of placing your order. Give us a try!

Borland Specials

Turbo C v1.1 (list ~~$100~~) **$59**
Turbo Pascal v4 (~~$100~~) *New Version!* **$59**
Pascal Editor Toolbox (list ~~$100~~) **$59**
Pascal Database Toolbox (list ~~$100~~) **$59**
Pascal Graphix Toolbox (list ~~$100~~) **$59**
Pascal Numerical Methods (list ~~$100~~) **$59**
Turbo Basic (list ~~$100~~) **$59**
Basic Editor Toolbox (list ~~$100~~) **$59**
Basic Database Toolbox (list ~~$100~~) **$59**
Basic Telcom Toolbox (list ~~$100~~) **$59**
Turbo Proglog (list ~~$100~~) **$59**
Prolog Toolbox (list ~~$100~~) **$59**

Prices good until December 31, 1987

Basic C
Turbo Pascal dBase
Publishing AI

C

Blaise **C Tools +** (list ~~$175~~) **$125**
Blaise **Turbo C Tools** for Turbo C (list ~~$129~~) ..... **$95** *New!*
Creative **Vitamin C** Specify compiler (list ~~$225~~) **$159**
Creative Programming **VC Screen** (list ~~$100~~) **$79**
Gimpel **PC-Lint** (list ~~$139~~) **$103**
Matrix **Synergy Development Toolkit** (list ~~$395~~) ... **$309** *New!*
Macintosh-like icons, menus, multi-tasking, mouse, more!
Metagraphics **Metawindow/All Language** (~~$195~~) ... **$159**
Windows, graphics, mouse support
Metagraphics **Metawindow/Turbo C** (~~$95~~) **$79**
Microsoft **C v5** w/Codeview & Quick C (list ~~$450~~) ..... **$279** *New!*
Microsoft **Quick C** (list ~~$99~~) ..... **$68** *New!*
Softfocus **"C" BTree & ISAM** w/source (~~$115~~) **$105**
Sterling Castle **Blackstar "C" Functions** (~~$129~~) ..... **$99** *New!*
Full featured library w/source for both MS & Borland
Unipress **Phact** RDBMS (~~$249~~) ..... **$199** *New!*
includes report writer, query program & all source

TURBO PASCAL

Blaise **Turbo Power Tools +** (list ~~$99~~) **$79**
Metagraphics **Metawindow/Turbo Pascal** (~~$95~~) **$79**
MicroHelp **Mach2 for Turbo Pascal** (~~$69~~) **$55**
windows, fast i/o, critical error trapping, more
Turbo Power **TDebug Plus v2** (list ~~$60~~) **$48**
Turbo Power **Turbo Utilities** w/source (~~$95~~) **$76**

dBASE

Bytel **Genifer** dbase III application generator (~~$395~~) **$225**
Comtel **dbScope** Interactive dBase debugger (~~$70~~) **$62**
Comtel **dbTools** (list ~~$70~~) **$62**
Fox **FoxBase + v2** fast dBase compiler (~~$395~~) **$249**
Tom Rettig **Library** specify dBase or Clipper (~~$99~~) **$79**
Wallsoft **UI Programmer** (list ~~$295~~) **$260**

OTHER LANGUAGES & TOOLS

Aldebaran **Source Print** (list ~~$97~~) **$74**
Aldebaran **Tree Diagrammer** (list ~~$77~~) **$59**
Digitalk **Smalltalk/V v2** (list ~~$99~~) ..... **$84** *New!*
Microsoft **Fortran** w/Codeview (list ~~$450~~) **$270**
Microsoft **Quick Basic v3** (list ~~$99~~) **$65**
MKS **Toolkit** Korn shell, vi, grep, 30 more (~~$139~~) **$115**
Periscope **Periscope II** (list ~~$175~~) **$139**
Phoenix **Plink-86 +** (list ~~$495~~) **$315**
Sterling Castle **Basic Development Tools** (~~$99~~) **$79**
B-Tree & windows w/source for MS & Borland basics
Texas Instruments **PC Scheme** great AI into (~~$99~~) **$77**

TSF carries hundreds of products from dozens of publishers. Call or write for a FREE catalog or a price quote.

Number 98 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Don't Knock IBM

I wish Steve Baker would refrain from his anti-IBM PS/2 propaganda. I used to use a Compaq 386; now I use an IBM PS/2 Model 60. I prefer the Model 60, and I will eventually buy a Model 80. I am developing a protected mode (80286) operating system, and I find the PS/2 better suited for such a system.

Shawn W. Barham
Garden Grove, CA

If I have a bias toward OS/2, it's only that I've been waiting so . . . long for it to arrive. And when it finally does, it may be too late. Many folks wonder whether the 80286 will survive the 80386.

I've always wondered whether the bugs in various steps of 80286 chips installed in older ATs have made OS/2 difficult to write. Will owners of the original IBM AT need to buy a new CPU chip, or will the operating system code around these problems?

From a distance, if I were looking at the high end operating system market on the 386, would it be easier to rewrite a single tasking DOS for multi-tasking, or would it be easier to modify an existing multi-tasking OS (like Unix) with a DOS-like shell?—MSB

Getting Out

Since I have an Okidata-92 printer, I was quite interested in the article by M. Steven Baker on a "Graphics Print Screen Utility" (Volume 5.2). In the process of getting this nice program to run, I found a minor typo:

On page 9

```
mov ax offset newint5
```

should be

```
mov ax, offset newint5
```

I also found that if the program was run twice, instead of gracefully exiting it hung up. I am using PC-DOS 3.3, so this may occur only with the new DOS, but I changed the exit as follows:

```
done_2
     mov ax,0   ;could set al to i
                ;if an ERRORLEVEL
                ;exit code is
                ;desired
     mov ah,4ch
                ;new line added--
                ;uses a different
                ;interrupt
                ;function to exit
                ;deleted "mov dx,
                ;ax"--since I don't
                ;use cpm86
     jmp bdos
```

These corrections allow a graceful exit when the program is inadvertently run twice.

I also found that the screen dump reversed black and white. To print unreverse the screen dump, I added one line as follows:

```
list_gr:
   xor     al,03fh      ;to
                        ;unreverse
                        ;video--
                        ;remove line
                        ;for reverse
                        ;video
   cmp     al,ETX
   jne     list_gr2
   call    listout
```

I'm pleased with the utility and hope you keep up the good work.

Dennis Revie
San Diego, CA

I should have mentioned the black on white reversal. The program for which this utility was originally designed draws white on a dark screen, so I casually reversed the output to the printer. Regarding the exit problem, I'm at a loss. I've not had any problems using DOS 2.1, 3.1, or 3.2. DOS 3.3 can be reali-ably run only on the new IBM PS/2 series. Apparently the soon to be released DOS 3.4 and 3.5 are mostly bug fixes.—MSB

Critique of Volume 5.5

Jon Greenblatt ("A Note on Optimizing Turbo Pascal") gives some ludicrous optimizations! What he is proposing would not be maintainable.

John Newlin ("TSRs Using Expanded Memory with Turbo Pascal") uses too much inline code, also a maintenance issue.

Ben Myers
Harvard, MA

More on Unorthodox Turbo Pascal

Re Jon Greenblatt's "Note on Optimizing Turbo Pascal": I have experience with Turbo Pascal, but I was totally lost with this article—too cryptic for me.

Richard Hendricks
Tewksbury, MA

Local Exchange

Hey! Your sales are going too well. I went to Software ETC., hoping to buy your latest issue, only to find they were sold out. That'll teach me. Next time, when I see it, I buy it.

I was very interested in your article on "Variable Passing and Hiding in QuickBASIC 2 Subprograms" (*PJ*, Volume 5.4, July/August 1987). Finally, a serious programmer's journal (no pun intended) that deals with advanced BASIC topics. Dare I ask for more on the subject? New implementations of BASIC have something to offer, I think, when the thoroughness of C or Pascal is not needed.

As for your journal, I'll keep on looking. In any case, keep the work up.

Michel Biedermann
Eugene, OR

Thanks for the good words. We try to keep a little BASIC in the magazine although we do get a fair number of complaints about it. There are those who share Wirth's opinion of the language. If there's an article or area of interest you don't want to miss, back issues are available. Be sure to bug your friendly local dealer to carry additional copies. Here in Eugene, PJ is also available at The Book Mark, Walden Books, and B. Dalton's. Subscribe to be absolutely certain you won't miss an issue. —Ed.

Recommendations

I really like *PJ*! My wish list of improvements: (1) Make it cheaper, or (even better) keep it at the same price and come out monthly; (2) run more C articles; (3) run an expert's

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

evaluation of several user interfaces in MS-DOS S/W; (4) perhaps run a series on XLISP (it now includes MS-DOS and graphics functions). This advice has a money-back guarantee.

Scott Zimmerman
Monrovia, CA 91016

We like the monthly idea, too, are shooting for sometime in 1988. In 1987 we're trying to grow within our bi-monthly publishing schedule—you'll notice this issue's bigger than the last. Tell us more about XLISP. —Ed.

Sweet and Sour Note

Until recently I didn't know *PJ* existed. I was delighted to see the quality of articles produced in your magazine—they are both informative and entertaining. It was because I was so impressed that I started advertising and submitting articles.

A sour note. It seems to me that magazines both large and small neglect the small time advertiser. My very first ads for a product called "The Pointer" from Stau Enterprises were placed in the Sept/Oct issue of *PJ* and the September 15th issue ofa large bi-weekly magazine. The Pointer is a memory resident help system and demo program generator. Many of my users say it's the best thing since sliced bread. Neither magazine's readers will know that because both ads were mutilated.

The idea I was trying to get across was lost because someone didn't take the time to read the copy properly. I spend a great deal of time preparing my ads because I'm proud of my work, and I want people to know it. Just because I don't spend millions of dollars in advertising like Borland doesn't mean my ad shouldn't receive just as much attention. After all, almost everyone starts out small, and if our advertising dollars are spent in vain, we will never be able to grow. This hurts everyone—the advertiser because he wasted his money, the magazine because a company that dies will no longer spend money on advertising, lastly, the consumer because there might be a product that they need and won't be able to get because they didn't see what they wanted in the ad.

I am very pleased with the way *PJ* is handling their mistake. You bent over backwards to correct your mistake, and I am loyal to you for life because you've treated me (a small advertiser) with respect. The other magazine, on the other hand, seems to feel that it's my tough luck, and they won't do anything about it. I just blew $500 bucks.

Paul Stauskas
President, Stau Enterprises Inc.
Mississauga, Ontario
Canada

We Hope We're Not Too Late

Please cancel—your magazine is less helpful to a programmer than *PC Magazine*. Plus, it takes the usual attitude of people who don't know any better about one of the best languages around—FORTRAN—you totally ignore it! That just tells me how ignorant you are. I've been programming for 26 years, and in my opinion, assembly, C, and FORTRAN are all equally useful, yet you ignore one of them. You do not deserve to call yourself "Programmer's Journal"!

Steven M. Hodge
Tacoma, WA

We think we should cover FORTRAN, too. Next issue, Steve Baker will look at the problem of porting mainframe software to the PC. In the course of his survey, Steve will compare FORTRAN compilers and introduce us to the Definicon 32-bit coprocessing engine.—Ed. ♦

"Mace Utilities 4.0 Best Data Recovery There is Gets Better"

PC Magazine
March 31, 1987

...And Better!

VERSION 4.10

RECOVERY FEATURES

DIAGNOSE checks any disk for errors.
REMEDY fixes errors, automatically.
UnDELETE recovers deleted files easily, automatically.
UnFORMAT recovers from an accidental format of your hard disk even when MACE was not installed.
new *Floppy* **UnFORMAT** recovers from accidental format of a floppy disk.
new ***FORMAT-F*** floppy formatter. Replaces DOS FORMAT. Includes **'RESURRECT'** option, to revitalize weary disks.
new ***FORMAT-H*** hard disk formatter replaces DOS FORMAT. Guaranteed safe.

HOT-ROD FEATURES

new ***Fast UnFRAGMENT*** quickly reorganizes segmented files for speedier disk performance.
VCACHE buffers data reads and writes in regular, expanded or extended memory for dramatic disk speedup.
new ***SORTD*** Directory sort.
new ***SQZD*** Directory path optimizer.
VSCREEN Screen speedup.
VKETTE Diskette caching for floppies.
new ***dbFIX***
Repairs and recovers damaged or lost DBase files, automatically.

Advanced performance at the same good price: $99

To Order Call
800-523-0258
503-488-0224

Paul Mace
SOFTWARE
400 Williamson Way, Ashland, OR 97520

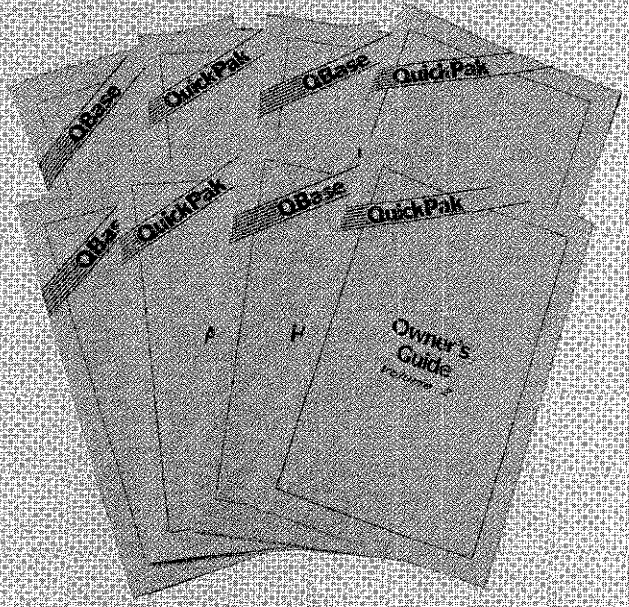
Number 80 on the Reader Service Card

PRODUCED 2024 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

BASIC Programming Tools

There's nothing basic about these professional programming utilities.

Whether you're a seasoned expert or just starting out, we can help you create programs that run faster, work harder, and simply look better. We have built our reputation on customer satisfaction by providing expert advice and quality technical support. All Crescent Software products include source code, demonstration programs, clear documentation, and a 30-day satisfaction guarantee.



- **QBase** *is a superb screen designer and* full-featured relational database. Because we include complete BASIC source code, QBase can be customized to suit your needs. Besides its database capabilities, QBase is outstanding for creating custom titles, help screens, and product demos, and includes a versatile slide-show program. **$99**

NEW!

- **QBase Report** enhances QBase by generating reports with multiple levels of sorting, automatic record updating, browsing, and transaction posting. As a special bonus, QBase Report includes a complete, ready to run, time billing application for computer professionals. **$69**

- **QuickPak** contains more than 65 essential routines for BASIC programmers. Included are programs for windowing, access to DOS and BIOS services, searching and sorting string arrays, creating pull-down and Lotus™ menus, accepting data input, and much more. QuickPak is loaded with examples and tutorial information, and comes with a clever tips and tricks book, plus *The Assembly Tutor* — a complete guide to learning assembly language from a BASIC perspective. **$69**

- **By Customer Demand — QuickPak II** More than 30 additional tools, including disk and printer tests to eliminate the need for On Error in your programs, and a multi-line text input routine that lets you put a note pad with full word wrap anywhere on the screen from within your programs. Other routines include binary file access, more menus, multiple screen save and restore, continuous time display, automatic box drawing, and much more. (Available for use with QuickBASIC *only.*) **$49**

NEW!

- **GraphPak** is an extensive collection of routines for displaying line, bar, and pie charts automatically within your programs. It will create 3-D charts with manual or automatic scaling, titles and legends in any size or style, as well as scrolling and windowing *in graphics*. GraphPak also comes with a sophisticated font editor for customizing your own character sets. **$69**

CRESCENT SOFTWARE

64 Fort Point Street, East Norwalk, CT 06855
(203) 846-2500

Separate versions are available for Microsoft QuickBASIC and Borland Turbo Basic — *please* specify when ordering. No royalties, not copy protected, of course. We accept Visa, M/C, C.O.D. and Checks. Add $3 shipping and handling, $10 overnight and foreign, $25 2nd day foreign.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

VEDIT PLUS

#1 PROGRAMMABLE EDITOR

Call 1-800-45-VEDIT for FREE Fully Functional Demo Disk

Stunning speed. Unmatched performance. Total flexibility. Simple and intuitive operation. The newest VEDIT PLUS easily satisfies the most demanding computer professional.

Try a Dazzling Demo Yourself.

The free demo disk is fully functional—you can try all features yourself. Best, the demo includes a dazzling menu-driven tutorial—you experiment in one window while another gives instructions.

The powerful "macro" programming language helps you eliminate repetitive editing tasks. The impressive demo/tutorial is written entirely as a "macro"—it shows that no other editor's "macro" language even comes close. And VEDIT PLUS is only 40K in size.

Go ahead. Call for your free demo today, You'll see why VEDIT PLUS has been the #1 choice of programmers, writers and engineers since 1980.

Only VEDIT PLUS is this Flexible.

The installation lets you pick from closely emulating the keyboard layout of Word Perfect, WordStar and others. Or you can easily create your own layout and even your own editing functions. Supports any screen size—you pick screen colors and attributes.

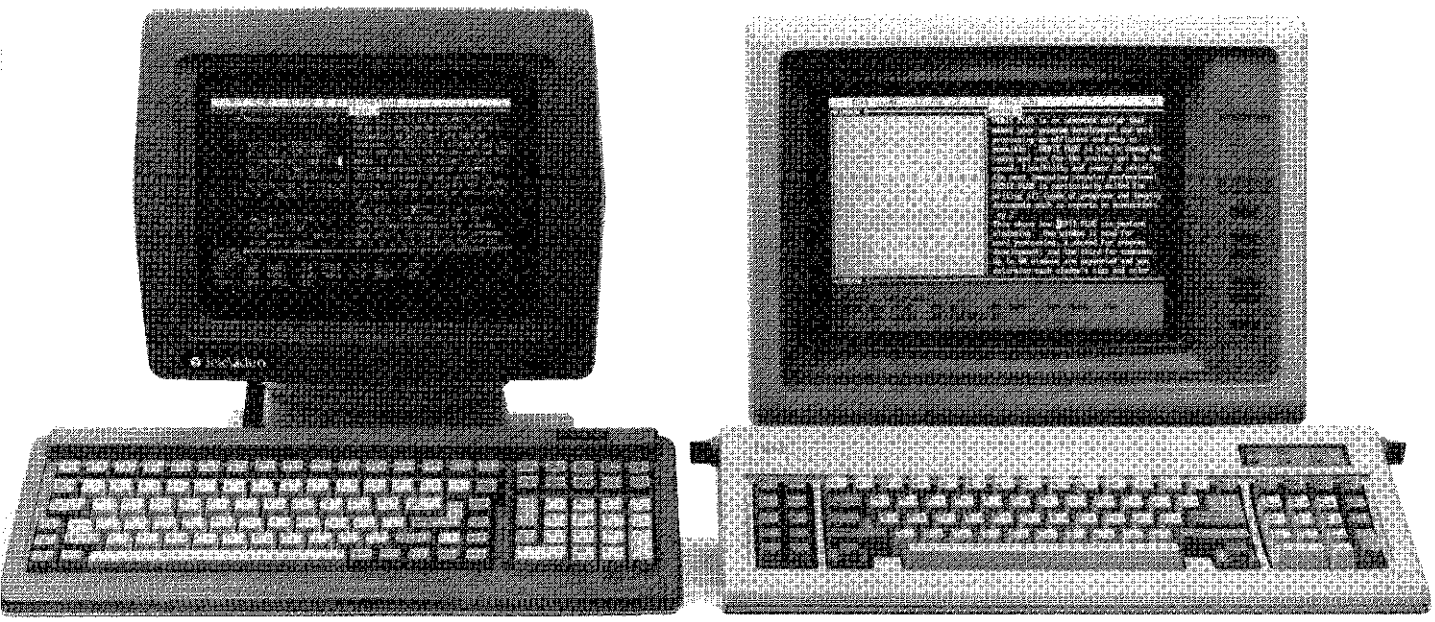
Supports the IBM PC, XT, AT and PS/2. Also supports MultiLink, PC-MOS/386, Concurrent DOS and most networks. Also available for MS-DOS, FlexOS (protected mode), CP/M-86 and CP/M. (Yes, we support windows on most CRT terminals, including CRTs connected to an IBM PC.) Order direct or from your dealer. $185.

Special: VEDIT (single file, no windows) for CP/M—$49.

- **Fully Network Compatible**
- **Call for XENIX-286 version**
- **30 Day Money-back guarantee**

Compare Features and Speed

	BRIEF	Norton Editor	PMATE	VEDIT PLUS
'Off the cuff' macros	No	No	Yes	Yes
Built-in macros	Yes	No	Yes	Yes
Keystroke macros	Only 1	No	No	Unlimited
Multiple file editing	20+	2	No	20+
Windows	20+	2	No	20+
Macro execution window	No	No	No	Yes
Pop-up menus	No	No	No	Yes
Execute DOS commands	Yes	Yes	Yes	Yes
Automatic processing of Compiler errors	Yes	No	No	Yes
"Cut and paste" buffers	1	1	1	36
Undo line changes	Yes	No	No	Yes
Paragraph justification	No	No	No	Yes
Convert to/from WordStar	No	No	No	Yes
On-line calculator	No	No	No	Yes
Configurable Keyboard	Hard	No	Hard	Easy
43 line EGA support	Yes	No	No	Yes
Manual size/index	250/No	42/no	469/Yes	380/Yes
Benchmarks in 120K File:				
2000 replacements	1:15 min	34 sec	1:07 min	6 sec
Pattern matching search	20 sec	Cannot	Cannot	2 sec
Pattern matching replace	2:40 min	Cannot	Cannot	11 sec



VEDIT and CompuView are registered trademarks of CompuView Products, Inc. BRIEF is a trademark of UnderWare, Inc. PMATE is a trademark of Phoenix Technologies Ltd. Norton Editor is a trademark of Peter Norton Computing Inc. MultiLink and PC-MOS/386 are trademarks of the The Software Link, Inc. CP/M and FlexOS are trademarks of Digital Research. MS-DOS is a trademark of Microsoft.

*Also available for TI Professional, Tandy 2000, DEC Rainbow, Wyse WY700 and others.
*Demo disk is fully functional, but does not readily write large files.

CompuView

1955 Pauline Blvd., Ann Arbor, MI 48103
(313) 996-1299, TELEX 701821

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The Second Coming—Paging and Segmentation, Multics, and OS/2

by
Christian Walker

Disneyland

All that glitters is not Jello.
Something is rotten in Disneyland.
Birds of a flock weave indiffering feathers.
Your ounce of prevention, enumerable powers,
climbs out of the frying pan into blunt embers.

A penny saved is a piece of museum.
Something is rotten in Disneyland.
Children need be heard. Are you listening?
A stitch in time saves enumerable needy,
one finder, one keeper, few merciful greedy.

"Curiosity," cooed the believers.
Something is rotten in Disneyland.
Don't cross your throats until they've exploded.
Do unto others. Bag not, nor sorrow.
I read these tomorrowland, tomorrow, and tomorrow.

—Don Russell

Controlling complexity is a driving motive of the computer industry. You can see this in hardware, software, and user interfaces. The best computer systems—like the best teachers—simplify life or make complex problems understandable. Unfortunately, another trend works against this tendency: Systems get bigger, slower, and more complex as users request or require more features or bigger data bases. Building systems within these two broad themes is what makes this industry interesting, chaotic, and rewarding.

Watching microcomputer systems develop in the context of these conflicting drives is like watching history as it's replayed in high speed mode. The early micros were not much better than the computers of the '50s. They were smaller and cheaper, but they lacked the software advances of the '60s and '70s. It didn't take long for micro systems to catch up. Take languages, for example. Assembly language was the language of choice at first, then BASIC, later came Pascal, and now C seems to be the darling of the industry. And Ada, that megalanguage that takes megabytes and MIPS to run and megahours to learn, is waiting in the wings.

Now, programmers don't run out and learn new languages just for the fun of it. They usually have some very good reasons. Features such as portability among machines, improved coding productivity, the support of reusable modules, and easier debugging often drive the decision to change languages. And with new, more complex languages, programmers tend to produce larger, more complex, and more (or sometimes less) reliable products.

Once you have new languages, you also have new functions. To support these languages and functions, as well as to become more competitive, companies frequently produce new hardware; for example, the Intel 8088 processor has an optional floating point processor that makes numeric computation much faster for very little extra money. Now a number of specialized processors do the work that used to be done largely with general purpose machines—the current batch of graphics processors clearly illustrates this trend. Generally speaking, when some reasonably common operation takes too much time in software, you will see a hardware advance to help solve the problem.

At the same time that hardware is produced to replace software, there is also a trend to produce hardware that allows something useful to happen that couldn't be successfully accomplished by software alone. Hardware paging and segmentation, for example, allow general purpose multi-processing to occur where only ad hoc solutions existed before.

Now before I get a flood of letters from software fans—"I have a multi-tasking operating system for the IBM PC"—let me say that my definition of "useful" is probably different from yours. A multi-processing operating system that lets me execute a HALT instruction, even if by mistake, or that allows me to overwrite another task's code or data areas is not my idea of a good time. This particular version of Acquired Immune Deficiency Syndrome can be deadly: I prefer dealing with one infected program or task at a time, not with an entire system of diseased programs that will become increasingly feeble over time.

It's natural that the micro designers at Intel, not wanting to leave any useful computer design of the last 40 years untried, would want to bring us more presents from history: in this case, the ideas of paging, segmentation, and protection. With just a few changes to the 8088 instruction set, the newer processors such as the 80286 and the 80386 can support operating systems that multi-process, page code or data, and guarantee security. With the hardware in place, it only remains to write the operating system to enable users to accomplish these tasks.

But writing a reliable multi-processing operating system is not easy. Don't expect to write one overnight, or even over a long weekend. Multics, one of the earliest examples of a paging and segmentation operating system, was written during the '60s. It took several years to complete and quite a few mil-

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

lion 1960s dollars, but even when it was finished, it still wasn't what you would call reliable.

Why Is It So Difficult?

Before you can understand what an operating system such as IBM's Operating System/2—which claims to support multitasking, multi-processing, paging, and segmentation—is trying to do, you should understand what the underlying hardware allows it to do.

Paging is a way of allowing a program to use a large address space while allocating only a small amount of real memory to it. When a program generates an address to code or data, the address will be translated via a system "page table" to its real address. If that address is not currently represented in real memory, the page table is marked so that instructions that try to access the address will be halted and the operating system notified of the request. The operating system can then locate the memory, read it in from the disk, fix the page table, and restart the instruction. Except for the extra time it takes to swap in the memory (and possibly swap some other page out since memory may be over-committed), the application program is not aware of anything's going on. This is known as "virtual memory" because the program can think it's using, say, 20 megabytes of memory even though the computer really has only 2 megabytes. The IBM System/370 machines use this simple technique.

. . . you realize that you have to manage, not 16 segments as you did on the IBM PC, but 8,192 of them.

Segmentation, unlike paging, must be visible to the application program. Instead of providing a single linear address space, segmentation allows the program to break up its data and code areas into separate address spaces. The advantage of segmentation is that it allows applications or languages to organize their different data sets more easily. Code can go in one segment, the stack in another, and other segments can hold tables and data bases that the program uses. With a reasonable OS, each segment can grow and shrink independently, without opening the Pandora's box of dynamic memory management. Finally, segments help catch bugs before they do too much harm. If a program indexes off the end of an array, it normally can start modifying whatever variables or code follow the array in memory. But if the array is in a segment, the processor will trap into the OS (or debugger) when it sees an address beyond the end of the segment. Segments can be made read-only (the code segment, for example), and memory belonging to other programs can be made completely inaccessible while your program is running.

To develop an address using segmentation, the program has to specify a segment and an offset within the segment. Each segment has its own offset zero, but the segments don't need to be the same size. There are several ways in which the segment/offset pairs can be specified. The segment number can be given in each instruction as well as the offset. Then the hardware looks up the base of the segment in a table and adds in the offset to get the address, which, if we also have paging hardware, gets sent through the address translation described above.

But a simpler way is the way Intel did it: Have a small number of segment registers that hold segment descriptors. A segment descriptor consists of the segment's base address, size, and access rights. Each instruction has only to specify a segment register and an offset. In Intel's case, the program loads the segment register with a segment selector, not an actual descriptor, which points to an entry from a system table of segment descriptors. The descriptor is then loaded into an invisible register. This way, the program has no way to generate addresses that the operating system considers out-of-bounds. This scheme forces the program to know when to load the segment register, but that's usually not too much of a hardship.

Intel took this straightforward scheme and added a few twists. The segment descriptor also has a "present" bit indicating whether or not the segment can be accessed. If the segment is not present, when the segment register is loaded the instruction causing the load will be halted and the operating system will be entered. In this way, one can write an operating system that "pages" whole segments at a time instead of using the page translation hardware.

To get more specific, let's look at the two Intel processors that support these features: the 80286 and the 80386. Both allow programs to run in the old style 8086 mode. In this mode, segment registers are not special and act just as they would on your IBM PC. This is the mode your IBM PC AT runs in if you are running MS-DOS. The mode that supports segmentation is called "protected mode." On the 80286, segment selectors can be up to 8K, and each segment can be as large as 64K bytes, so the program's address space can appear to be 512 megabytes large—which isn't bad until you realize that you have to manage, not 16 segments as you did on the IBM PC, but 8,192 of them. Oh, and by the way, there is no paging hardware: The operating system has to page whole segments at a time.

The 80386 in protected mode has 32-bit offsets, and there are 64K segments. Each of these segments can be up to 4 gigabytes long! The program's address space, therefore, is 256K gigabytes, which should keep most programs busy. The

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Unbelievable!

SOURCER™

Creates commented source code and listings from memory, .COM files or .EXE files

- **CLARIFY UNDOCUMENTED CODE**
- **EASILY MODIFY PROGRAMS**
- **RECREATE LOST SOURCE CODE**
- **DOCUMENT FOREIGN CODE IN ENGLISH**

SOURCER™ creates detailed commented listings and source code directly suitable for assembly. Built in data analysis and simulator resolves multiple data segments and provides detailed comments on interrupts and subfunctions, I/O ports and much more. Supports 8088 through 80286, and V20/30.

BIOS SOURCE

PS/2 ▪ AT ▪ XT ▪ PC ▪ Clones

- **CHANGE & ADD FEATURES TO YOUR BIOS**
- **CLARIFIES BIOS INTERFACES**
- **SPECIFIC TO YOUR MACHINE**

The bios pre-processor to SOURCER™ provides the first means to obtain accurate legal source listings for any bios! Identifies entry points with full explanations. Resolves PS/2's multiple jumps for improved clarity. Provides highly descriptive data labels such as "video_mode" and "keybd_q_head," and much more.

SOURCER™	$ 99.95*
SOURCER™ with bios pre-processor	$139.95*

(*OUTSIDE USA, ADD $15 SHIPPING; CA RESIDENTS, ADD SALES TAX)

To order or receive additional information just call!

800-538-8157 x 811 (outside Calif.)
800-672-3470 x 811 (inside Calif.)

V COMMUNICATIONS
3031 Tisch Way, Suite 200, Dept. J1
San Jose, CA 95128
(408) 296-4224

PS/2, AT, XT, PC are trademarks of International Business Machines Corp.

Number **125** on the Reader Service Card

program could, of course, restrict itself to just one 4 gigabyte segment and avoid the agony of constantly reloading segment registers. And paging hardware is available if you don't want to swap whole segments (remember they can be up to 4 gigabytes long) to the disk.

What Is the Point of All This?

Some advantages were mentioned above—virtual memory that's larger than the real memory of the computer and the simpler memory management that segments provide. In addition, a properly written operating system can use this hardware to easily protect programs, to keep one program from trashing another, or to protect sensitive data or code from being modified or copied. Other hardware related to the segmentation can also prevent programs from entering operating system code at the wrong place or at all. All of this can lead to a more reliable, trustworthy system, easier to understand and program for.

The Multics system pioneered operating systems of this ilk. The designers made mistakes that are well known. (It must be nice to work at a place that gives you brownie points for admitting to your mistakes.) A few that come to mind:

1. Multics had only 1 megabyte per segment, which hurt it competitively by the end of the '70s. Never restrict your segments to a small address space.
2. Having state of the art features on a system that crashes every 2 hours isn't a big help. Reliability is more important than features.
3. The Unix system is based on the principle that processes are easy to start and communicate with. Processes should be simple to create, not time consuming.

It is my hope that the designers of OS/2 do not make the same mistakes. Unfortunately, they have already picked the 80286 as the processor of choice, which means they are already making mistake number one. The 80386 would be a far, far better base for an operating system that will last well into the 1990s.

But Who Knows?

Maybe IBM will be perverse and repeat its mainframe history. Then we can look forward to seeing several IBM operating systems, all with different advantages and disadvantages, all with different interfaces to the operating system so that programs will have to change, all with different user interfaces so that our customers will pick the one of their choice. Then I know I will never have to worry about job security: I can spend the rest of my life reading IBM (or Microsoft) manuals describing the lastest changes to a set of what we hope are working operating systems. ♦

Christian Walker is a founder and vice president at True BASIC, Inc. He has been responsible for the IBM PC versions of the True BASIC language systems, which also run on the Apple Macintosh, the Amiga, and the Atari ST. He is currently supporting his opinion of the 286 by porting True BASIC to an 80386 running in protected mode. Send constructive comments to

True BASIC, Inc.
39 S. Main Street
Hanover, NH 03755

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The Pointer

The Memory Resident Help System and Demo Program Generator

The Pointer is the ultimate tool for creating powerful demo programs and creating help systems, all without programming! Here are just some of the features that are too numerous to list.

Help Features:

- Save hundreds of hours creating help systems for your programs
- Retro-fit old applications with a new help system in just minutes
- All commands are menu-driven for fast execution and easy learning
- Simply move the cursor to the word you want help for and press enter. A window will pop-up instantly with your help information
- Supports mice for fast cursor movement and menu selection
- No programming involved
- No source code generated for you to compile and link into your program
- No limit to the size of help system generated
- Runtime version requires only 68K

Demo Features:

- No limit to the size of your demo
- Create demos that:
  - are self-running
  - require user interaction
  - interact with and drive a live application program
  - combine all of the above
- Hand holding feature lets you create demos that will drive a live application, not just a picture show like those other demo programs
- Demos can be run from DOS or from inside your program
- Supports mice for effortless creation of windows and pointers
- Windows can be any size, up to a full screen
- Ideal for creating tutorials, demos and for prototyping

Best of all, you can create as many help systems and demos as you like. NO ROYALTIES EVER! Money Back guarantee.

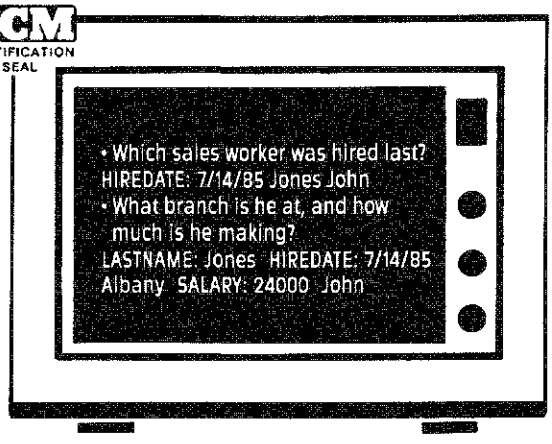
ORDER NOW! Only $129.95 U.S.
(416)-270-6673
Stau Enterprises Inc.
3650 Kaneff Cres., Suite 1808, Mississauga Ont. Canada., L5A 4A1

STAU ENTERPRISES INC.

Number **134** on the Reader Service Card

PCM CERTIFICATION SEAL

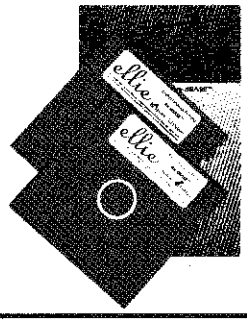
NY residents add applicable sales tax.


• Which sales worker was hired last?
HIREDATE: 7/14/85 Jones John
• What branch is he at, and how much is he making?
LASTNAME: Jones HIREDATE: 7/14/85
Albany SALARY: 24000 John

dBASE, dBASE III are trademarks of Ashton-Tate Corp.

true natural language data access for dBASE

ellie: *English Language Frontend*
• not copy protected • only $189
• for PC/XT/AT • dBase III/III Plus
• 512K min. RAM • Info: 212 316-9078
To order, send check/money order:



ELFSOFT
210 W 101 NY NY 10025

Number **113** on the Reader Service Card

Software Developers

We need your program!

Do you have a program that's good enough to sell, but don't want the problems or financial risk of producing, typesetting, printing, packaging, warehousing, marketing, distributing and supporting a product?

Why start your own software house?
We've done it for you!

Merlin Publishing Group
is now accepting submissions of micro-computer software for publication.

You get:
- initial cash payments
- generous royalties
- to spend your time programming

But we can't help you if you don't submit. Call or write today, for our submission guideline kit.

MERLIN PUBLISHING GROUP ™

1240 Johnsons Ferry Place, Suite A10
Marietta, GA 30068
(404) 977-6034

P. S. See us in Atlanta at COMDEX-Spring '88

Number **141** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Report from the Usenix Conference and Pink Noise

by
Frank D. Greco

Apparent at the last Usenix conference in super-hot Phoenix, Arizona: PostScript will become an integral part of future windowing systems; NeWS is a truly innovative piece of software from Sun; X Windows needs help; and Steve Jobs has good taste in clothes.

[If you never heard of the Usenix organization before, it is a highly technical user group that gets together twice a year to share software and hardware ideas and discoveries relating to the Unix world. Usenix is similar in many respects to the ACM SIGGraph or ACM SIGPLAN organizations.—FDG]

Can't Hack It

Much to the dismay of many in the techie crowd, former Apple head Steve Jobs delivered the Usenix keynote "Why We Have to Make Unix Invisible." Sporting an expensive-looking European suit right out of *GQ* magazine, Jobs started off by saying that he was not as technical as his (tee-shirted) audience, that he was not a "Unix kernel hacker" and did not know "the difference between ls and dir." This struck me as a good tactic. It made the audience feel less aggressive and allowed him to make some strong statements about technical topics without some hacker's jumping down his throat. Clever.

It'll Be in Tomorrow's Paper

A consummate publicist, the chairman of both NeXT and Pixar companies, Jobs then proceeded to drop several quotable bombshells regarding Unix and Unix workstations. The first was "Unix must become a mainstream operating system by 1990, or it will die." He then compared the small number of total Unix installations (250,000) to the huge number of Macintosh (1,000,000) and MS-DOS (6,000,000) installations. Heedless of the validity of his ultimate argument (i.e., Unix has to become invisible), most in the highly technical audience chuckled as Jobs compared a multi-user OS with single-user OSs. Certainly, if the comparison is to be more toe-to-toe, Unix's numbers must be multiplied by the average number of users per installation. This could be anywhere from one to 100, with a likely average of four to five users per Unix installation. Anyway, why is being "mainstream" (whatever that means) an indication of success?

I'm on Your Side

Jobs then compared a small roster of companies for and against Unix: AT&T, Sun, Apollo, and the universities for; and IBM, DEC, Apple, and Microsoft against.

He explained that the four major companies against Unix have substantial business interests in their own operating systems and applications. They do not want Unix to succeed. IBM has VM (and others), DEC has VMS, Apple has the MacOS, and Microsoft has MS-DOS and OS/2. Of the four trying to make Unix succeed, Jobs said that many universities and Sun Microsystems are doing an excellent job; that AT&T, however, does not know what it is doing with Unix. This AT&T remark elicited many laughs and hearty applause. By way of demonstrating this point, Jobs noted the recently announced merging of AT&T's Unix with Microsoft's Xenix and added, "AT&T is shooting themselves in the foot that way." It's fairly obvious that Microsoft wants OS/2 to succeed, not Unix. If you really believe that Microsoft is going to give a lot of attention to this merged AT&T/Microsoft Unix, think again. "There's lots of money riding on Unix being unsuccessful," Jobs concluded.

PS, I Love You

As far as Unix windowing systems go, Jobs took a deep breath and said, "I believe X is brain-damaged and will die" (this from someone who self-admittedly does not know the difference between an ls and a dir). Jobs said that MIT's X Windows is kludgey and not the way to go. Referring to future workstations (NeXT workstations?), Jobs said, "Graphics and windowing will be PostScript-based." He then paused and muttered in a low voice, "but maybe not NeWS. . . ." So it is fairly obvious that the NeXT workstations will indeed use PostScript in their windowing system—in an approach similar to Sun Microsystems'. Having PostScript in both the graphics/windowing model and the printer subsystem insures true WYSIWYG synchronicity between the screen and printer. When Jobs was at Apple, he was instrumental in adopting PostScript from a fledgling Adobe for the Apple LaserWriter.

The NeXT Recipe

Jobs closed his keynote by listing some requisites if Unix is to become "mainstream" and thus successful:

1. Have only one variety of Unix.
2. Have PostScript graphics/windowing.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

3. Have a user interface for mere mortals.
4. Have *no* system administration.
5. Evangelize third party vendors.
6. Produce a low-cost, breakthrough Unix computer.

Just like a guest plugging his newest record or TV show on David Letterman, Jobs was subtly alluding with his last point here to his new baby, the NeXT workstation.

A Head-First Slide?
Jobs concluded the keynote with a slide that read, *"Cu Unix estos la Esperanto de la kumputila industrio?"* This translates into, "Is Unix the Esperanto of the computer industry?" I thought it was a little unimaginative and corny, but . . .

There Was Plenty of Jobs Here
Compared to many other talks at the conference, Steve Jobs' speech was very professional albeit not quite so technical (remember that this *is* a Usenix conference). It appeared to me that Jobs was trying to promote his company through his keynote (I know, I know, that's no big revelation). After all, many of the top Unix gurus in the world were at the conference, and some wizard might be attracted to Jobs' new venture. I noticed a few analysts from Wall Street there, besides, and it doesn't hurt to keep yourself in the public eye—especially if you eventually decide to take your company public. After the talk, the obviously charismatic Jobs held an informal audience with a semi-circle of about 200 to 300 people.

And Now, for the Rest of Our Show
The rest of the conference was a mixture of tutorials, talks, and exhibits. The exhibit hall and the number of exhibitors were (for a Usenix conference) typically small, but there were a fair number of interesting technical talks on such topics as multimedia editors, graphical debuggers, the Mach OS, multiprocessor Unix, and streams implementations. Two papers on the Mach operating system from Carnegie-Mellon University were particularly interesting: "A Unix Interface for Shared Memory and Memory Mapped Files Under Mach" and "Mach Threads and the Unix Kernel: The Battle for Control." Keep your eye on Mach as a model for future operating systems.

X On? Nope, X Off!
As for the tutorials, I attended both the X Windows and NeWS lectures. X Windows is a networked window system developed and copyrighted by MIT's Project Athena. Contrary to the popular impression, it is not public domain; however, it is distributed freely. The X Windows tutorial was given by two gentlemen from DEC. It was atrocious. The handout material was just a listing of all the X Window subroutine calls—literally. If I want to see what the X Windows call for mouse control, color query, or checking for window events is, I'll look it up in the manual; you don't need a tutorial for that! There were minimal source code examples (two one-page examples), no X or generic windows background, no benchmarks, and no comparisons with other window systems. The instructors were rookie speakers (and obviously very green instructors).

X Marks the Spot
At one point during the X tutorial, Marc J. Rochkind, the author of *Advanced Unix Programming*, asked about six times whether the endpoint pixel of an X-drawn line was turned-on when the line-drawing primitive was executed. It became a joke on how many permutations a question could take before the lecturer could understand the question. The question was never answered, indicative of the entire day. The X tutorial was a complete waste of time.

Sun Gives Good NeWS
The NeWS (Network extensible Window System) tutorial on the other hand, was first-rate. It was superbly conducted by members of the staff of Sun Microsystems. The handout material from Sun for the NeWS tutorial was excellent—even additional information and sample source code along with hard copies of the visuals.

David La Vallee of Sun's Technical Sales started off with an excellent overview of both NeWS and PostScript and how they both fit into Sun's scheme of things. La Vallee opened with Sun VP and co-founder Bill Joy's: "PostScript is the ASCII of today." Considering the intense emphasis on graphic environments lately and the richness of the PostScript model, this remark does indeed make a lot of sense.

Sun's Owen Densmore discusssed programming for the PostScript-based NeWS. Apparently, at least until Sun gets their CPS (C language to PostScript filter) completed, you should be proficient with PostScript to program for NeWS. The benefits are impressive. NeWS is a highly-programmable, networkable, and extensible window system. It also incorporates and heavily uses fine-grained multi-tasking, that is, lightweight processes. This multi-threaded computation within one (heavyweight) process is similar to the internal tasking that Modula 2, Concurrent Pascal, and Ada all have. This is how NeWS achieves such high performance despite the many complaints that printer-PostScript is extremely slow. Although the tasking in NeWS is currently non-preemptive, preemptive multi-tasking is planned for future versions of NeWS.

You can have overlapping windows, tiled windows, pull-down menus, pop-up menus, slide-across menus, even arbitrarily sized windows (not necessarily rectangular!) on any machine in a network. You can even move the mouse pointer from one workstation's screen to another through the network capabilities of NeWS. Lawsuits notwithstanding, you can even emulate any other window system with NeWS. It's all in how you program the NeWS "kernel" (actually, "server" is more correct). You

"Is Unix the Esperanto of the computer industry?"

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

download the desired window system attributes via PostScript programs (NeWS programs) into the NeWS server, and off you go. And if you write well-behaved graphics programs for published interfaces for window systems such as Microsoft Windows, the Macintosh graphics interface, Sun's SunView, or even GEM, it is conceivable that one day soon, all could be running simultaneously on the screen of one workstation. Not that it is very useful to the everyday PC or workstation user, but the fact that this is potentially do-able is fascinating.

One NeWS programmer I talked to during a break made a funny, striking comment regarding the ability of NeWS to display arbitrarily-sized windows. "Someone asked me why would I ever want non-rectangular windows.... Hell, I don't know, but when I find out, I'll let you know." NeWS gives you all the flexibility in the world; it is like the C language in many respects.

One obvious application for non-rectangular windows is for pop-up menus. Normally, when a pop-up (or pull-down) window appears, it is drawn with a shadow to give it the illusion of depth. The systems programmer has to therefore design his menus with an extra graphic (the shadow). In NeWS, a pop-up window can be programmed to have a shadow; it is formally part of the window. The programmer no longer needs to drag in extra baggage just for a pop-up window; it is already there.

X-Rated NeWS
The NeWS tutorial came to an end with someone in the audience claiming that NeWS was losing industry momentum to X Windows. A Sun representative said that Sun has always been a proponent of industry standards and has joined the X Windows industry consortium. Sun will offer a combined X.11/NeWS product whenever the X.11 specifications are finalized at MIT. Programs that write for the X protocol and X library will work in this new product. Companies that use Sun's X.11/NeWS product will get the best of both worlds—industry standards and a flashy, highly extensible and state-of-the-art window system. I'm sure it will be offered at a reasonable price; the standard NeWS executable lists for a mere $100!

Even More Jobs
Most of you know by now that H. Ross Perot, former president of Electronic Data Systems, Inc. (EDS), has already invested 20 million dollars in Steve ("I don't know ls from dir") Jobs' NeXT venture. Jobs is using 12 million of his own dollars along with 1.3 million dollars from Stanford and Carnegie-Mellon universities to start up NeXT. Perot has been quoted, "I saw significant technological breakthroughs at NeXT. They have some truly revolutionary products under development." The strong rumors from the Sun Microsystems booth at the Usenix exhibition hall were that the NeXT workstation was going to be 68030-based and that Jobs bought quite a few Sun 25 MHz 68020 boards. I also overheard some hackers (one had a NeXT nametag) mention that NeXT appears to be keenly interested in the new 10 MIPS (those are VAX MIPS folks) RISC-based board from Sun.

Andy's Gang
Another hot rumor: One of the founders of Sun Microsystems, quiet (and filthy rich) Andy Bechtolsheim, who actually built the SUN 1 (Stanford University Network 1) box, has started a wholly-owned subsidiary of Sun. Apparently, he took two of the top Sun hardware engineers with him. No one knows exactly what this hardware wizard is up to, but someone at the Sun booth smiled from ear-to-ear and told me, "Just wait—you'll see."

fopen() Says Me
I had an interesting conversation with a fellow Usenix attendee about using optical disk file servers in a network. A few people are implementing Sun's Network File System (NFS) drivers for optical disk media. NFS is implemented using Sun's Remote Procedure Call (RPC) and eXternal Data Representation (XDR) mechanisms. Using NFS ensures that the interface semantics between the optical disk and the application program are identical to other programs that access any other disk drive. There are no extra library or system calls to learn; you just **fopen()** or **open()** any file on the optical disk drive as you would any other file. Obviously, this makes life easier for all the software engineers out there.

Sony Sono Buoni
I talked to a few hackers in the exhibition hall who are working on a prototype of a Sony erasable optical disk system for PCs. According to these techies, Sony will announce the disk system in the fall, probably at Comdex. The optical disk drive will cost around $6,500, and each disk will dance to the tune of approximately $200. The disks themselves hold 350 megabytes per side and have an average access time of 100 milliseconds. Remember the old IBM XT's hard disk drive's access time? It was 85 milliseconds. These Sonys are almost as fast!

A Postscript on PostScript
With all the talk about PostScript, I asked Ram Rao, a technical representative from DEC who is intimately familiar with X, about Steve Jobs' comments on X Windows. Ram said "X Windows is a technology of today, and NeWS is a technology of tomorrow." Ram told me that X Windows will also use PostScript. Adobe is working closely with DEC and IBM on this one; however, PostScript will be used only as a printer output filter and not as a screen graphics model. Tsk, tsk . . .

Port Whine
Unisoft demonstrated their unadorned, vanilla Unix for the Macintosh II and said it was not for sale. It was only for demo purposes. Apple has received the Unix port from Unisoft and is currently interfacing the Macintosh ROM with Unix themselves. According to Unisoft, they asked Apple if they could do the complete Macintosh port including the ROM interfacing. Apple turned them down. It seems that Apple does not trust anyone with the source code to their ROM.

Pink Noise

Resolution—Number Nine
The Pepper SGT graphics board combines the capabilities of both Intel's 82786 coprocessor and TI's 34010 graphics microprocessor for under $1,000. It can emulate the CGA, MDA, or EGA standard video modes. Number Nine Computer Corporation from Cambridge, Massachusetts, has combined both graphics processors onto a single board upgradable to 4 megabytes of video RAM. In text mode, the Pepper SGT uses a crisp 16 x 19 character cell. Similar to the newer VGA modes, the SGT can display (on a multi-synchronous monitor) 640 x 480 pixels in 256 colors, or 800 x 600 (or 1280 x 480) in 16 colors. All colors are selected from a monstrous palette of 16-Meg colors! Another key feature is the ability to allocate the video memory into multiple bit-maps of programmer-selectable size, resolution, and color. Since this board has much higher bit-map addressability

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

than screen resolution, Number Nine also offers hardware zoom, smooth panning, and smooth scrolling. Talk about flexibility! The vendors are also claiming that they have drivers for Windows (i.e., PageMaker), GEM (i.e., Ventura), and Lotus.

Here Come the Res, Here Come the Res

Speaking of resolution, Univision Technologies of Burlington, Massachusetts, has an Intel 82786 graphics coprocessor-based graphics board capable of displaying 2048 x 1536 x 8-bit (that's 256 shades, folks!) on the new Tektronix GMA-201 high resolution display. This is going to be great for imaging applications. Unfortunately, it costs $6,995; however, these guys have a cheaper board that has 1280 x 1644 x 8-bit for $1,595 that is perfect for electronic publishing applications.

O.K. for A. Kay

Seiko looks close on announcing a 14-inch color LCD display with 640 x 440 resolution. Because of the natural "flatness" of the display, this unit would make an interesting display for some transportables. This may also help Apple's Alan Kay reach his Dynabook vision.

Fiber LAN

No, it's not a rug store. Several vendors are targeting the upcoming Fiber Distributed Data Interface (FDDI) standard with hardware and software products. Chip manufacturer Advanced Micro Devices (AMD) in Sunnyvale, California, has already announced a 5-chip set to support the new standard, which will support nominal data rates of 100 megabits per second. A 100 Mbps fiber-based LAN is capable of transmitting all 48,000 pages of the *Encyclopedia Britannica* from one node to another in 13 seconds! National Semiconductor and Texas Instruments also appear close to announcing support hardware. FDDI will allow the industry to tie several applications such as video, voice, and data using a single communications medium: one big step towards the elusive ISDN.

DEC's Net Profit Will Not Result from a Token Effort

The ANSI committee for the FDDI standard, in case anyone wants to know, is X3T9.5. Since DEC is very interested in Ethernet technology and has been criticized for not supporting a "real-time," deterministic networking protocol such as token-passing, they are extremely interested in a 100 Mbps Ethernet network. In a super high-speed network, the non-deterministic CSMA/CD or CSMA/CA (Carrier Sense Multiple Access/Collision Detection or Collision Avoidance) Ethernet standard takes on pure deterministic characteristics (i.e., real "real-time"). Expect several FDDI-based Ethernet products from DEC in the near future.

Excuse Me While I Squish This Guy

If you are involved in sending large bit-maps across a network, you realize the tremendous bandwidth required of the LAN. If you don't have a super high-speed network and must send bit-maps on a LAN, consider the AMD 7971 (Advanced Micro Devices 408 / 732-2400), which will compress simple (two-color) bit-mapped images that are up to 16K pixels per line wide. That implies that it can handle documents that are 3.4 feet wide and 400 dpi. The compression algorithm is based on the standard CCITT groups III and IV techniques for facsimile equipment. AMD claims compression ratios of 50:1; that is probably for a very sparse bit-map. From my experience in this area, I would guess a 20 or 30:1 ratio is about average. That would mean that a black and white 2048 x 2048 bit map, which is about 4 megabits (524 K bytes), would be compressed to about 140K bits (about 18 K bytes) on average. It looks perfect for bit-map oriented distributed CAD/CAM applications.

This Is a Recording

Oki Semiconductor (408/720-1900) recently announced a solid-state version of an audio tape recorder. The MSM6258 speech processor has on-chip memory controllers, voice-detector circuits (to avoid recording dead air), A/D and D/A converters. The RAM controller can remember the starting addresses of up to seven recorded messages for random retrieval. The first versions of this chip can store up to 8.6 minutes of actual speech. Imagine when Nolan Bushnell gets wind of this one!

Wyse He Here?

Interesting questions and a lot of conjecture were raised when president and CEO of 3Com Corporation William Krause was named to Wyse Technology's board of directors. Wyse knows that the future is networking, and they obviously see the need to incorporate more communications capabilities into their products. It is well known that 3Com has tremendous expertise in local

DAN BRICKLIN'S DEMO PROGRAM ONLY $74.95

Read what they're saying about this popular program for prototyping and demo-making:

"A winner right out of the starting gate. After you use DEMO once, you'll wonder how you got along without it."
—PC Magazine

"Everybody who writes software, either commercially or for in-house applications, should immediately order a copy. Period. No exceptions."
—Soft·letter

Product of the Month
—PC Tech Journal

Thousands of developers and most of the largest and best known software companies are using this program. You can, too. Act now!

NEW TUTORIAL! JUST $49.95

The perfect companion to the Demo Program. The Tutorial helps you learn the ins and outs of its basic and advanced features. Complete with a 96 page manual containing step-by-step instructions, diskette, and function key template.

ORDER NOW!
1-800-CALL-800 x8088

VISA MasterCard AMERICAN EXPRESS

Use 800-number for orders only.
Questions, special shipping, etc., call **617-332-2240.**
No Purchase Orders. Massachusetts residents add 5% sales tax. Outside of the U.S.A., add $15.00.
Requires 256K IBM PC/Compatible, DOS 2.0 or later. Supports Monochrome, Color Graphics, and EGA Adapters (text mode only). The Tutorial requires the Demo Program.

SOFTWARE GARDEN, INC.
Dept. J
P.O. Box 373, Newton Highlands, MA 02161
Number 108 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

vLIB

the user interface library for C

A comprehensive, easy to use library of custom C functions for building sophisticated PC applications.

Windows • *overlapping, tiled, built-in window manager*

Menus • *vertical, horizontal ("Lotus" style), pulldown ("Mac" style)*

Forms • *single call input and display, multiple field types*

Pop Ups • *messages, prompts, selection lists*

Full Editing Input

High Speed Display

... and much, much more!

Give your programs a professional look! Develop applications faster!

Compilers supported: Microsoft C and Quick C, Borland Turbo C, Lattice C, and Datalight C.

All memory models supported.

Demo disk with usable sample programs, source, and reference card . . . **$10**

Library with reference card and detailed programmer's manual **$99**

Library (as above) with full source code **$149**

Prices include shipping, CA residents add 7% sales tax.

No royalties. Site license available.

Call or write today:

Pathfinder Associates
291 Madrone Avenue
Santa Clara, CA 95051
(408) 984-2256

Visa and MasterCard accepted.

Number **145** on the Reader Service Card

area networks. I would guess that the next generation of Wyse PCs or workstations will all include a 3Com-designed Ethernet transceiver, just as today's machines have a serial port. I would not be surprised if the San Jose Wyse-enheimers have a large-screened 386 diskless node running OS/2 in the labs. In addition, since Wyse now owns Amdek, which has a tremendous distribution network, these machines have the potential to be very successful. I'm sure that the inclusion of Mr. Krause on Wyse's board of directors will only help the ever-growing Wyse Guys.

Black and White and Read All Over

I was talking to a laser engine technician for a large foreign conglomerate the other day about what print engines are popular with OEMs and users. . . . It seems that there are two basic types of laser printer engines: the "write white" and the "write black." The "write black" types, e.g., the current Canon, use the laser to trace the black portions of a page. These types are good for mostly text pages. The "write white" (say that 10 times fast!) types, e.g., the Ricoh, use the laser beam to trace out the white portions of a page. This type is much better for dense graphics. The "write white" types seem to be winning more support nowadays as they produce blacker blacks on the page.

From talking to other laser printer engineers I gather that, despite squeezing 600 dpi out of the older Canon laser engine with some software sleight-of-hand, the latest revision of the Ricoh engine is the current pop hit.

This LSD (Large Storage Device) Passes the Acid Test

Honeywell (303/773-4581) recently announced a new high-density tape unit that stores 5.2 gigabytes on ordinary (camera-grade) VHS tape. Its sustained recording rate is 4 megabytes per second, and its recording density is 50 megabits per square inch. Additionally, you can search the tape in granularities of blocks of 32K byte chunks. Cost? A measly $44,000.

Knowledge Is Good

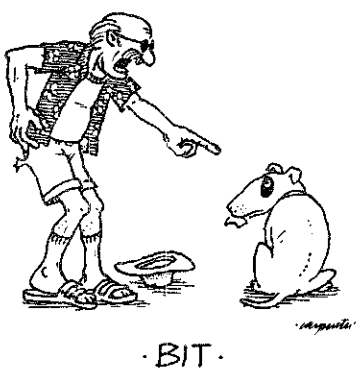
A colleague recently told me of some interesting research regarding relational databases and Unix done at his alma mater in Britain. Apparently, someone replaced the Unix filesystem with a relational database. While still offering the same system calls, this system's flexibility offers the programmer and end user inherent advantages such as increased querying, browsing, and analysis of the data. The next step of this research was to incorporate a knowledge base instead of a relational database.

"Readers Respond" Department

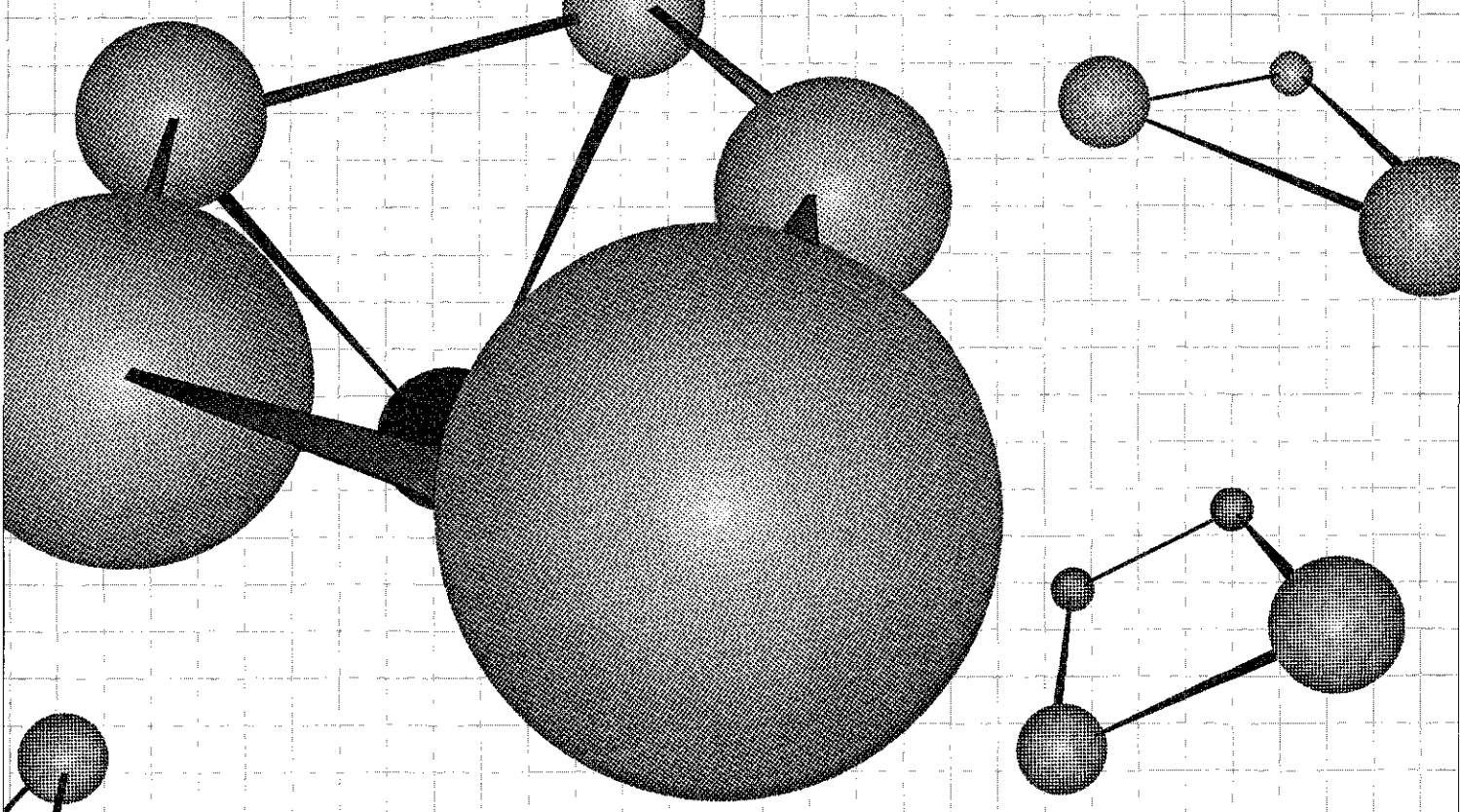
A few readers pointed out a minor bug in my "Redirection, or They Went That-a-way" article (*PJ*, Volume 5.1). My comments in the assembler routines GETSW.C (line 17) and SETSW.C (line 22) should refer to the AL register and not the AH register. Sorry about the typo. Also, George Erlanger of Colebrook, Connecticut, asked why I didn't use MSC 4.0's **intdos()** call and stay "within C" instead of writing two assembler routines. You're 100% right, George. I could have written both routines using **intdos()**. [Mr. Erlanger was kind enough to send me the source of a little program that does exactly that.—FDG] I was just being pedagogical. I merely hoped to show certain *PJ* readers how to interface assembler routines to Microsoft C. Besides . . . my two assembler routines take up less space than **intdos()**, *and* I really cannot agree that using **intdos()** is "staying within C." It's a heavily operating system and machine dependent library call. Thanks for the calls and letters, everyone. ♦

Frank D. Greco is an extremely busy independent consultant who specializes in highly usable computer systems. Readers can contact Frank at the following address:

Frank D. Greco
Crossroads Technologies, Inc.
P.O. Box 530
Fanwood, New Jersey 07023
(201) 754-7820



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



Aztec C - the *best* C in the Micro-cosm

Why is Manx Aztec C the best C ?

Because Manx offers a combination of performance, features, and pricing that are unbeatable.

What kind of performance ? In the 2/87 issue of Computer Languages, our 3.2 release of Aztec C for PC-DOS turned out 109,489 iterations of the Dhrystone benchmark per minute to108,303 for Microsoft C. Our 4.1 code is even faster.

What kind of features ? Features like in-line assembly code, a professional make utility, library source, ROM support, debuggers, assemblers, and overlay linkers. Many of these features are unavailable or extremely high priced options in other systems. Microsoft, for instance, does not provide library source or in-line assembly.

What kind of pricing ? For $199, the Professional version of Aztec C gives you an Optimized C, a macro assembler, a librarian, overlay linker, extensive UNIX and MS-DOS specific library routines, a powerful debugger, sample programs, and over 500 pages of documentation. Manx Aztec C has a hard earned and well deserved reputation as a leading C software development system for PC-DOS and MS-DOS. At this price, it is also one of the best software values available.

Over 80 percent of our sales are repeat sales or are based on recommendations. That says it all. At Manx, we extend ourselves to the limit to provide the best software, the best service, and the best support possible. Call today, and join the over 50,000 users who know the Manx difference.

Manx also provides C systems for Macintosh, Amiga, CP/M-80, TRS-80, Apple //, 6502/65C02 ROM, 8080/Z80 ROM, 680x0 ROM, and 8086/80x86 ROM. **Call for prices and information.**

Aztec C86 4.1

PC-DOS • MS-DOS • ROM

Aztec C86-p Professional System $199

• optimized C with near, far, huge, small, and large memory • in-line assembler • in-line 8087/80287 • ANSI support • Fast Float (32 bit) • optimization options • Manx Aztec 8086/80x86 macro assembler • Aztec overlay linker (large/small model) • source level debugger • object librarian • MS-DOS 3.x file sharing & locking • comprehensive libraries of UNIX, MS-DOS, Screen, Graphics, and special run time routines.

Aztec C86-d Developer System.....$299

• includes all of Aztec C86-p • Unix utilities make, diff,grep • vi editor • 6+ memory models • Profiler.

Aztec C86-c Commercial System.....$499

• includes all of Aztec C86-d • Source for library routines • ROM Support • CP/M-86 cross dev. option • One year of updates.

How To Become An Aztec C User

To become an Aztec C user call **800-221-0440.** From NJ or international locations call **201-542-2121. Telex: 4995812** or **FAX: 201-542-8386. C.O.D., VISA, Master Card, American Express, wire (domestic and international), and terms are available.**

Aztec Systems bought directly from Manx have a 30 day satisfaction guarantee. Systems are upgradable by paying the difference in price plus $10. One and two day delivery is available for all domestic and most international destinations.

MANX

Manx Software Systems
1 Industrial Way, Eatontown, NJ 07724

To order or for more information call today.

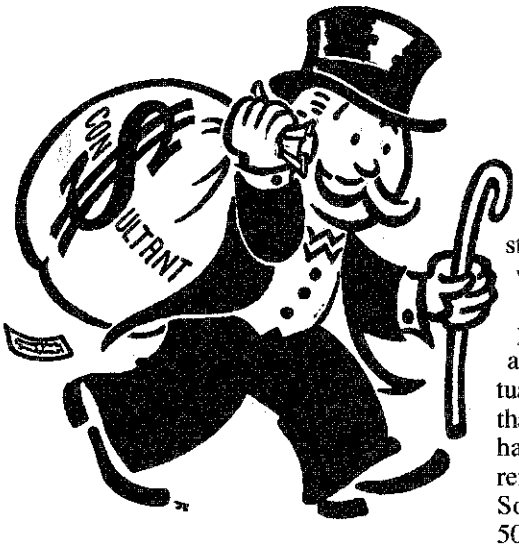
1-800-221-0440

In NJ or international call (201) 542-2121 • TELEX: 4995812

REPRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

How to Get Rich in Consulting

by
Paul Barkley



When someone offers to tell you how to get rich quick (for the measly sum of perhaps twenty dollars), the answer is often the act itself—find someone else gullible enough to give you money to tell *them* how to get rich. That's the way it works. You don't need to be a philosophical economist (or an economical philosopher) to suspect there's something wrong with a theory that purports to make everyone rich.

After quite a few years of independent consulting, it has finally become apparent to me how one maximizes one's income in computer consulting. I should have figured this out a long time ago, but I guess I'm a slow starter. I'd like to sum up this insight in one word, the way they do it in your usual get-rich-quick articles. Unfortunately, the issue is complex and the answer is multifaceted, so we'll look at six ways to increase your income. If these methods won't make you rich, at least they should help make you more financially comfortable.

The More You Produce, the More You Make

If you want to make more money, all things being equal, all you need to do is produce more product or service for the same amount of overhead. The easiest way to execute this principle in independent consulting is to work more hours.

If you accept small jobs from clients (as opposed to long contracts) and perform only part of the tasks on-site, you are in a position to adjust the total number of hours you work. Instead of working 40 hours per week, try working 50 or 60 hours per week. Whatever you bill, you can increase your annual revenue 50% by working an additional 20 hours per week. In actuality, your net may increase far more than this since the first 40 hours per week have to cover your overhead—office rent, utilities, marketing, and so forth. So, if you increase your billable time 50% without any increase in your overhead, your net will rise dramatically—much more than 50%.

Here's an example: Let's say you pay $10,000 a year for overhead and take in revenues of $40,000, leaving a net of $30,000. If you increase revenues 50% to $60,000, your net increases to $50,000, which is a 67% increase.

Consulting is one of the few businesses where sales can increase without costs also rising. As you probably know, there are two kinds of costs. The first is fixed costs and has nothing to do with your level of sales. The rent you pay is the same whether you bill $1, or $100,000. The second kind of cost is "variable." If you put 50 cents' worth of material and 50 cents' worth of direct labor into each widget you manufacture, each widget costs you $1 in direct costs. The more widgets you make, the more variable cost you incur.

Although there's no such thing as a truly fixed cost (rent goes up in increments as a firm grows into gradually larger quarters), most costs for a consultant certainly act fixed in the short run; thus, any increase in sales has a dramatic impact.

There's a proverbial fly in this financial ointment, though. In order to increase your billable hours, you must already have work lined up and waiting. Many consultants, particularly in the early years, don't have enough work to cover the basic 40 hour work week. (For a drastic solution to too little work, see the "Draconian Solution" at the end of this article.) In fact our example above showed a starting gross of $40,000, which could be 1000 hours per year at $40 an hour (perhaps mostly programming), or 2000 hours at only $20 per hour.

Increase Your Hourly Rates

One obvious way to make more money is to increase your hourly rates; thus, by working the same number of hours, but charging more per hour, you make more. Very simple.

The problem here, of course, is that most businesses find that their clients are not pleased with price increases. Economists measure this market resistance to price change using a concept called "elasticity." Some products (and services) are relatively price-insensitive. Customers don't care. In those situations, the obvious strategy is to increase prices in order to maximize net income.

Most products and services are more sensitive to price. As you increase price, the demand for your service falls. Some potential customers decide that they simply can't afford the service at all while others turn to your more reasonably priced competitors.

Price elasticity is an interesting concept. Over small ranges, demand for a service may not be affected by price increases; thus, you can safely raise your price a modest amount and get a corresponding increase in net income.

The tricky part, of course, is finding out what your services are worth in your market. Try to find out what your competitors charge. If possible examine competitive bids for this information. (In many bidding situations, you're entitled to see the other bids after the award has been made.)

You can probably make more money than you currently are by raising your hourly rates a little. Still, this has the potential to cost you a great deal of business, and as you'll see in the discussion of the next principle, hourly rates should

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

be abolished wherever possible.

Structure Work to Avoid Hourly Rates

Doctors. Dentists. Lawyers. Jewel thieves. Many professions are noted for the high earning power of their practitioners, but there's a trap here for the unwary consultant. These professionals may earn high hourly rates, but their total earnings are limited to what each can accomplish "with his or her own two hands" in a typical work week. There's a flat upper limit, regardless of the hourly rate.

These professions also suffer from the same problem consultants face: You just can't bill a full 40 hour week and handle all the other business tasks as well. Many of these professions routinely operate with support staff in order to maximize the total return by billing the principal's rate to clients and paying far lower rates to get the nursing/hygienic /clerical tasks accomplished. Consultants can do that, too, assuming they want to grow a business that has employees who not only support the business but who in turn must be supported, regardless of revenue and cash flow.

There's a better way for the consultant who wants to remain small. Stop charging fees on an hourly (or daily) basis, and begin charging fixed fees for identifiable deliverables.

Stop charging fees on an hourly basis, and begin charging fixed fees for identifiable deliverables.

Some tasks, such as analyzing and developing systems, don't lend themselves to fixed fee arrangements. There's simply no way to limit the scope of the project—the client might be thinking about all sorts of interesting extensions, particularly if the extra work involved is at an incremental cost of zero. Your revenue is fixed, but somehow your level of effort is not, and the client expects effort to rise to infinity. You almost have to perform this sort of task on an hourly, albeit high hourly, basis.

Once an analysis and design have been completed, however, there's no reason why you can't turn in a fixed fee bid to write the code and the user manual and perform the installation and training. It may take you some time (it took me years—maybe I'm a slow learner) before your estimating skills are sufficiently honed that you can make accurate bids, but once you learn this skill, you can greatly increase your income.

There are two components to this fixed fee business. First, you must track how long it takes you to write menus, screens, and reports, and to perform all the other odds and ends of creating systems, such as setting up file structures. Take careful notes.

Next, you need to develop tools that will allow you to speed up this process. You need to be able to produce a report that would normally take four hours to code by hand, in perhaps an hour. Now you have three hours of "free" time to work with. Similarly, find ways to speed up other coding and documenting tasks. You want to strive for high quality work with an absolute minimum of hours on your part. Reliable tools will not only produce initial code faster, but they will also virtually eliminate many kinds of routine debugging chores. You have to get the code right only once, and then you can use it repeatedly.

When you bid a fixed fee job, you may be able to assume that your competitors are not as automated as you are, that they'll have to code the old fashioned way. If so, bid the full hours, or a little less, and reap the additional profit by delivering a quality system (on time) with a disproportionally small effort on your part.

Keep in mind that your entire work psychology changes. You're on the hook if anything goes wrong, so you need to pay close attention to what you're doing. You also have the incredible motivation to finish as soon as possible. Ever read Dostoevsky? (Yeah, they made me read this guy in school, too. In a fair competition, he'd lose out to Harold Robbins any day.) In *Crime and Punishment* the prisoners begged the guards to "set them a task" for the day. When told what the task was, no matter how large, the prisoners would bust their tails to finish. When they were done, they could rest for the remainder of the day. Great motivation. On regular days, with no fixed task, they took all day to mosey through the work, even with the threat of physical punishment. Think about it in *your* work.

Think, too, about how other professionals may apply this principle. The best example is a law firm that takes a case on a contingency fee basis, often contracting to take a third of the award. With multi-million dollar awards, some firms find themselves with disproportionately large returns. (Of course, if they lose, they get nothing, so we don't want to dwell on this analogy too long—most consultants would not want to gamble with their incomes.)

To increase your net income, switch as many hours as you can from an hourly rate to a fixed fee basis, and find ways to increase your efficiency.

Find a More Profitable Clientele

It's not always possible to choose your clientele. I always feel grateful when someone from an unknown company calls me for help. Seldom do I turn work down.

Still, as a consultant begins to acquire a work backlog, certain clients and certain industries begin to look less desirable. Do you really want to go through the hassle for the amount of money that they bring in?

Identify your most profitable clients. What makes them special? What industry are they in? How can you tailor your services more directly for this target market?

Here's an example. I noticed over a period of time that law firm clients, for all their faults, seemed to offer a good return on my time. They're used to viewing time as a valuable resource measured in fractions of an hour. Since they charge

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

MetaWINDOW
Power Graphics for your PC!
PC TECH JOURNAL
"Product of the Month"
"... a technological tour de force for fast PC graphics."
NO ROYALTIES!
MetaWINDOW is ... portable ...
NEW! - QuickWINDOW/C
All the features of MetaWINDOW for Microsoft Quick/C! - $95*
... and window managers.
Unparalleled Performance!
MetaWINDOW provides an expanded set of graphic drawing functions, plus the added functionality and performance required for designing multi-window desktop applications.
• auto-cursor tracking
• pull-down menus
• pop-up windows
• comprehensive graphic functions
• multiple fonts
File
New ^N
Open
Close
Save ^S
Quit ^Q
10 Point 12 Point
Bold Italic
Enhanced Features!
• Display multiple bitmap or "filled-outline" fonts.
• Face fonts for bold, italic, underline or strike-out stylings.
• Full "RasterOp" transfer functions for writing, erasing, rubberbanding or dragging: lines, text, icons, bit images and complex objects.
• Create pop-up menus, windows and icons.
• Supports IBM's new PS/2 VGA and MCGA graphics.
MetaWINDOW comes complete with language bindings for 20 popular C, Pascal and Fortran compilers, plus dynamic runtime support for over 50 graphics adaptors and input devices.
MetaWINDOW
Advanced Graphics Toolkit
4 disks, 3 260 page manuals - $195*
NEW! - TurboWINDOW/Pascal
All the features of MetaWINDOW for Borland Turbo Pascal Ver. 4! - $ 95*
*Plus $5.00 shipping and handling
TO ORDER CALL 1-800-332-1550
For information or in CA call 408-438-1550
METAGRAPHICS
SOFTWARE CORPORATION
269 Mount Herman Road
Scotts Valley, CA 95066
Number 143 on the Reader Service Card

that way, they don't mind paying the same way. In fact, if your rates are too low, you can't even get their attention. (To get a hint, find out what law firms in your area are billing per hour for their new hirees right out of law school. Charge them that amount—an attorney will figure that that's about right for a "second class" profession such as computer consulting.)

For me, law firms are an attractive target, and publications aimed directly at this target market allow marketing to be focused. Choose your own desirable targets.

Go for Maximum Employment

Another principle is that if you want to maximize your income, you need to achieve a high ratio of billable to unbillable hours. This can be accomplished in a number of ways, but often this means *lowering* your rates, exactly the opposite of one of the previous principles.

If life were a textbook, we could plot a curve showing the relationship between the hourly rate charged and the number of hours per week that clients would want our services. In general, the more we charge, the fewer number of hours we bill. We noted before that there may be points where we can increase our rate without decreasing the demand at all. In those cases, the more we charge, the more we make.

Many consultants, though, are not getting enough hours to make a decent living. Although there may be a number of reasons for this, such as inadequate marketing or marketing that simply hasn't been sustained long enough (say the consultant is new in the field or hasn't been beating the bushes properly), another reason may be that there aren't enough clients who can afford the consultant's relatively stiff hourly price.

Try setting a lower price for some jobs, but *only* if you tie this to longer periods of employment; for example, if you charge $60 an hour, offer a client $40 an hour for a four week (160 hour) contract. If you can accomplish this 12.5 times a year and achieve full employment, you'll gross about $80,000 a year.

Many consultants use brokers to find jobs that last one to 12 months. The hourly rate is lower, but it's essentially full time. We won't go into the ramifications here of Section 1706 of the Tax Act, but longer term contracts will almost assuredly require that you accept the title (and other trappings) of an employee status. As a fellow consultant said, if a person waddles like an employee and quacks like an employee, he or she probably is an employee. Period.

Some of the most successful consultants I know have large, on-going contracts that provide substantial floors for their businesses. They contract for about half time—either 20 hours a week, or perhaps two weeks a month. This pays the overhead and, as one consultant puts it, "feeds the cat." The remainder of their time is spent on short, high rate jobs. Together they total full time employment.

Cut Your Expenses

In classic business situations, there are two ways to make more money. You must either bring in more revenue with the same costs (which we're already discussed), or you must keep the revenue steady while lowering expenses.

I've said on a number of occasions that I don't think much of running a consulting practice from a residence. It's just plain unprofessional. Yes, I've done it before, and I didn't like it. I did everything imaginable to make sure that no one came by to drop anything off.

In the last year I've mellowed somewhat because I started thinking about how else I could use the office rent money. Let's say your office costs you $400 a month. That's about right for a small office in most metropolitan areas, assuming you can find one to fit a one-man business.

What else could you do with the rent money? Well, the same $400 will make the monthly payment on a ten year construction loan to put a $30,000 addition on your house. In fact, that addition could be partly for office space. Sounds tempting, doesn't it? The same amount of money will also make the monthly payment on a five year loan for a $25,000 sports car (with a down payment of $6,000). From an investment standpoint, if you're even considering the car over the house addition, you need to go back to Finance 101. On the other hand, if the car doesn't have any appeal to you, you're probably spending too much time soaking up a green tan (or maybe you're unnaturally or prematurely sensible).

Looked at this way, an office costs a lot. It's not so much the actual monthly cost—rather, it's what a person can do with the money instead that makes it seem expensive.

When I first began consulting, I quickly became very frugal. It wasn't my innate personality; it was a necessity based on cash flow. Every time I'm tempted to buy some new piece of

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

hardware or software, I put it off for a week or two, and in many cases (if I'm lucky), the urge passes. Every dollar you can avoid spending, particularly on a recurring basis, is one you can save and invest.

The Draconian Solution: Drop Out

We've talked about the following principles:

- The more you produce, the more you make;
- increase your hourly rates; structure work to avoid hourly rates;
- find a more profitable clientele;
- go for maximum employment;
- and cut your expenses.

Taken together, these principles suggest a comprehensive plan of attack, albeit a drastic one: Obtain a full time employee position with some flex time, close your regular office and move it into your residence (thereby cutting expenses), and in your private practice do more fixed fee work in fields of your choosing, at higher rates.

Easy for me to say, right? Wrong. I decided that if this was such a smart move, I ought to try it out. I found a major project that was irresistible. I applied and was accepted as an employee of a large firm at an attractive salary. Although I finished all my private work before I began and didn't plan on doing much more than maintenance for my private clients, I should have anticipated that all those proposals I had pending with clients and potential clients would of course now be accepted. That's a basic law of business, related to the old saw that banks want to lend money only to people who don't need it. Business never arrives when you need it, but shows up in volume when you're overbooked.

Suddenly, I have more work than I know what to do with. My employer gets a full 45 hour week from me (I don't want any complaints from them). My private clients call my residence (not my new job site) and talk to my answering machine. I call them back as soon as possible, which sometimes is the next morning. (Very few, if any, of my clients have projects so time-sensitive that response in 24 hours is not satisfactory. I don't do any on-site support anyway.)

My expenses have almost vanished, and even health and life insurance costs have evaporated as the new employer assumes these costs. If you're an employee now, you may take that for granted, but for someone who's spent 20 years without a job, this is a minor miracle (along with paid holidays and regular paychecks, both bizarre but wonderful ideas).

I now have not only full employment, but another 20 hours a week of prime contracts as well (65 hours is about my limit). I can't tell you what an incredible difference all this makes in the bottom line—I'm almost appalled at the cash flow. (Please don't send me suggestions of how you can help me out with this embarrassing money problem—I'm working diligently on getting rid of it right now.)

Is this a cop-out? Have I forsaken independent consulting?

I think this depends on your attitude. I treat my employer as just another client. I've always maintained the attitude that I'll build anything my clients want, even if it's a real bad idea. When I suggest maybe seven more efficient ways to do something, but assure them that I'm just more than glad to do it any way they want, clients always seem to come around to my way of thinking—it doesn't challenge their authority or suggest that they're meatheads. As they weigh the costs, they almost always reach the intelligent decision. If not, it says that there are other important (hidden) decision criteria that they're not articulating.

Similarly, I tell my employer when I think we should be doing something differently, and when they cheerfully ignore my advice, I still give them the best work I can produce, regardless of how flawed the design is. That's not my problem, and I have a blizzard of memos to prove it. (I never knew how much fun memos could be—nobody reads them, but they can be entertaining to create and handy to have in the file.) It's all a matter of perspective, and besides, I often find out later that there were many hidden facts that affected the decision. All I can do is operate with the best information I have.

This constant disagreement with management is precisely what drives many people away from regular employment to independent consulting, where they are finally free—to starve. I've read countless messages on bulletin boards where disaffected employees long to be free from the tyranny of employers. Their "servitude" seems to eat at their souls, and they can't wait to be in a position where people will finally listen to what they say.

At any rate, I'm finding the merger of employment plus independent consulting to be a delightful combination,

MISOSYS

SOFTWARE FOR YOUR P.C.

Order Now 800-MISOSYS (647-6797)

ED/ASM-86™

- 80X86 Assembly Language development

An integrated editor, assembler, linker, and debugger for 8086, 80186, 80286, and 8087 assembly and disassembly. All features are in a single EXE file.

- a full-featured line EDITOR with intra-line editing, block move and copy, partial save and load.
- a full-screen turbo-type editor is supported on CGA-equipped systems.
- the ASSEMBLER supports structured coding; can directly generate a COM or EXE file; assembly to memory for faster debugging.
- The DEBUGGER includes symbolic disassembly of arbitrary code to disk.

ED/ASM-86 [M-86-030] . . $124.95 + $5 S&H

RATFOR-86™

- Rational FORTRAN translator

A professional implementation of RATional FORtran by James Beard, Ph.D. Provides structure and greater portability to FORTRAN programs to reduce your programming time and effort dramatically over that required when writing directly in FORTRAN. Includes tutorial reference manual.

- compiler has extensive macro support
- "arith" macro for binary arithmetic
- short form READ and PRINT
- enhanced switch case support
- Requires a FORTRAN compiler

RATFOR-86 [M-86-73] . . . $99.95 + $5 S&H

MISOSYS Inc.
P.O. Box 239
Sterling, VA 22170-0239
703-450-4181 MC, VISA, CHOICE
800-MISOSYS/647-6797

VA residents add 4½% sales tax.
S&H: Canada add $1;
Foreign use S&H times 3

Number 121 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

blending the best of both worlds, and I think all full time consultants who are not satisfied with their present income should consider this. I would suggest trying to secure a half or three-quarter time contract, though, and not full employment, unless you're fortunate enough to find a project that you *really* want to work on (as I did).

If you're already working full time, though, and thinking about getting into consulting, you have a much different problem. Of course, you can try for part time contracts while you continue to work. The problem is that you'll meet potential clients at a very slow rate because you won't be free to do much effective marketing. In addition, jobs will be too far apart for your experience base to accumulate other than very slowly.

I know a nurse-midwife who chose to work in an extremely busy urban birthing center/hospital in lieu of a smaller private practice because she felt she needed the experience. This center does about 20 births a day while many practices might do that many in a month. Her experience is accumulating at a rate almost 30 times higher than it would in private practice, and because of the high risk population served, she's seeing situations weekly that might occur only once in an entire "normal" career.

The analogous situation for the budding consultant is to take a full time position with a major consulting firm and work on as many projects as you can. When you have some experience, then go out on your own. If that's not possible, jump into consulting full time—don't try to learn the trade on a part time basis. It's risky, but necessary. Oh yes, plan on starving for a while—this will build "character" (and remove unwanted flab).

If you're consulting now, and you're not satisfied with your income, consider one or more of these "get rich" methods. You don't have to execute them all—just set the wheels in motion for any of them that appeals to you.

Oh, and by the way, do yourself a favor and invest the extra cash flow instead of spending it. If you do this long enough, you won't have to work at all. You'll be able to retire and live a life of leisure. ♦

Send your comments or questions to

Paul A. Barkley
The Barbican Company
6845 Elm Street
Penthouse One
McLean, Virginia 22101

If you need a

FAST C COMPILER with 8087 & software floating point and a full STDIO library with DOS 3 support
FAST ASSEMBLER that is stand-alone, or inline with no separate assembly step
FAST LINKER that makes stand-alone .EXE files supporting memory or disk based overlays
SOURCE DEBUGGER for both C and Assembler code
FULL SCREEN EDITOR that can edit 2 files with no size limits and can invoke the Compiler and Linker
GRAPHICS package that supports text, line, and bitmap operations

Then try

DeSmet DC88

We've combined our popular DeSmet C88 Development Package with the DeSmet D88 Debugger and our Graphics Package. Library source code is available separately.

$99

C WARE

P.O. Box 428
Paso Robles, CA 93447
Phone:(805)239-4620
Telex: 358185
BBS: (805)239-4834
We accept VISA, MC & AMEX
Call now and we'll ship today
945 Spring St #14, Paso Robles, CA 93446

Number 38 on the Reader Service Card

You have been *waiting* for this!

Turbo Instant_Compile T.M.

A new utility for Turbo Pascal programmers.

Drastically reduces the tedious and repetitive recompilation of previously debugged source code.

Ideal for programmers who appreciate Turbo's programming environment but who:
- Tire of watching Turbo's line numbers flash by the screen
- Fail, occasionally, to get everything entirely right the first time
- Use a library of purchased or previously developed code
- Believe computers should be adapted to serve humans

Instant_Compile saves your time and sanity by eliminating the need to repetitively compile perfectly good code in order to reach the portion you are developing. It is easy to use. You place a compiler directive in the source code. When this point is reached, the entire state of the compilation is saved. Your next compilation can ***instantly*** return to this point and proceed. Simple to use.

Now it is practical to use Turbo Pascal in developing larger programs. Requires: 256K IBM PC or compatible and Turbo Pascal 3.01A or 3.02A. Why wait? Call now!

(817) 654-5400

Introductory price: Only $49
100% satisfaction or your money back.

Synergy Systems T.M.

200 West 7th St. #613 ● Ft. Worth, Texas 76102

Number 150 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

EVEN MORE POWER AND FLEXIBILITY

BRIEF 2.0

Users and industry press alike have unanimously proclaimed BRIEF as the best program editor available today. Now, the best gets better with the release of BRIEF 2.0.

Straight from the box, BRIEF offers an exceptional range of features. Many users find that BRIEF is the only editor they'll ever need, with features like real, multi-level Undo, flexible windowing and unlimited file size. But BRIEF has tremendous hidden power in its exclusive macro language. With it, you can turn BRIEF into your own custom editor containing the commands and features you desire. It's fast and easy.

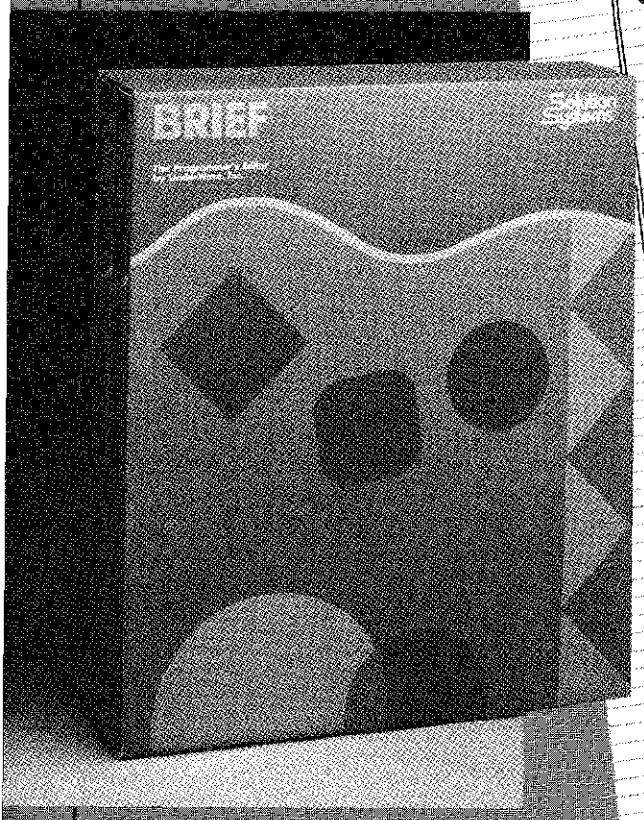
Jerry Pournelle, columnist for BYTE magazine summed it all up by saying "BRIEF is: Recommended. If you need a general purpose PC programming editor, look no further." His point of view has been affirmed by rave reviews in C JOURNAL, COMPUTER LANGUAGE, DR. DOBB'S JOURNAL, DATA BASED ADVISOR, INFOWORLD AND PC MAGAZINE.

One user stated "BRIEF is one of the few pieces of software that I would dare call a masterpiece." Order BRIEF now and find out why. BRIEF 2.0 is just $195. If you already own BRIEF, call for upgrade information.

TO ORDER CALL: 1-800-821-2492 (in MA call 617-337-6963)

As always, BRIEF comes with a 30 day money back satisfaction guarantee.

Solution Systems™
541 Main Street
Suite 410
So. Weymouth, MA 02190
(617) 337-6963



Look at these BRIEF 2.0 enhancements!

Main Features:

- All new documentation with tutorials on basic editing, regular expressions and the BRIEF Macro Language.
- Setup program for easy installation and configuration. (Requires no knowledge of the macro language)
- Increased speed for sophisticated operations like Undo and Regular Expression Search.
- Expanded regular expressions, with matching over line boundaries.
- More block types, with marking by character, line or column.
- Command line editing (move cursor, add and delete characters, specify command parameters).
- Support for more programming languages.
- Optional borderless windows.
- Enhanced large display support, including wider displays.
- Reconfigurable indenting for C files (supports most indenting styles).

Plus the basic features that made BRIEF SO popular!

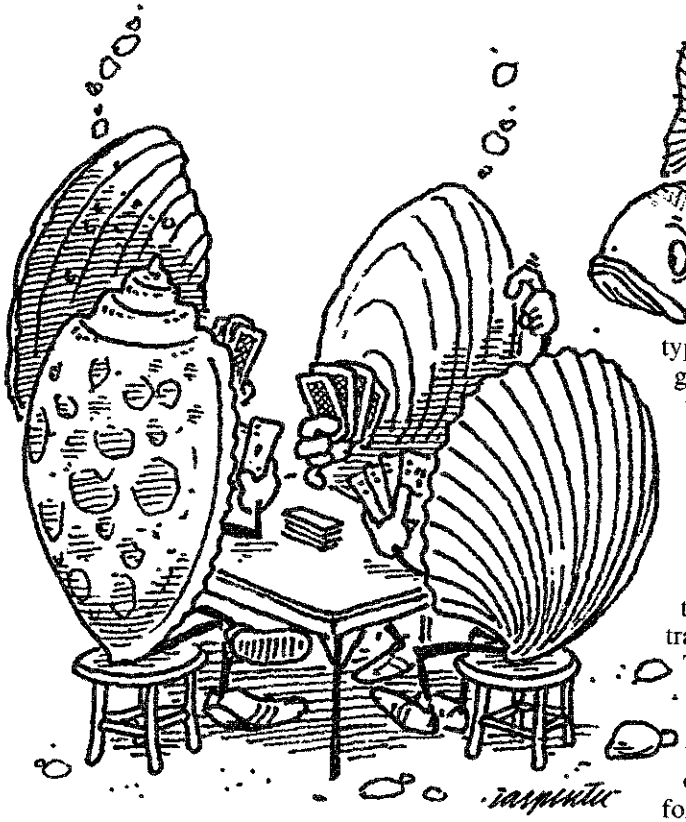
Basic Features:

- Full multi-level Undo
- Windows
- Edit many files at once
- File size limited only by disk space
- Automatic language sensitive indentation

IBM PC is a registered trademark of International Business Machines Corp.
UNIX is a trademark of UNIX Systems Laboratories.
Solution Systems is a trademark of Solution Systems.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The Shell Game



by
John J. Newlin

When I began the project of putting together a DOS shell for this article, I was tempted to name it YAD—Yet Another DOS Shell. Indeed, there are well over 30 commercial, Shareware, and public domain DOS shell programs currently available. I suspect that one reason there are so many shell programs is that they are so popular. Another is that shell users' tastes vary widely and users are prone to trying variants until they find one that suits them.

Writing a custom DOS shell for yourself can be very rewarding, particularly if you don't have a great deal of experience with DOS level programming. In addition to creating a useful utility that operates according to your specifications, you gain valuable high and low level programming experience, which leads to the natural question—just what is a DOS shell, anyway?

A DOS shell is primarily an interface between a PC user and the rich set of DOS system level functions that are not included in the standard DOS interface. No one would argue with the observation that most DOS commands are clumsy, anemic, and labor intensive. Microsoft, quite wisely, I think, chose to provide a minimal, spartan interface between its operating system and the user, thus leaving the task of designing and implementing a more powerful, more ergonomic interface to the independent programmer. And that is what this programming project is all about. The shell presented in this article is just that—a shell. For practical reasons, it is not complete in that it does not include all the bells and whistles most code developers and/or users would desire. But it does provide a framework upon which to build, and it solves, for the less experienced programmer at least, some of the more difficult problems of implementing a DOS shell.

The Programming Tools

Before getting into the code and how it works, I want to mention programming tools. Programming tools are second in importance only to the programmer's creativity and should be selected according to the type of project being undertaken. I selected Marshal Language Systems Marshal Pascal for this project for several reasons. One is that the Marshal system supports linking of separately compiled and assembled modules, a definite plus for systems level programming. Another is the speed and compactness of the code it produces. It is my belief that no other compiler for the PC, in any language, produces code that is as fast as the Marshal compiler's. Turbo Pascal programmers should know that by the time they read this, Marshal will have released their Turbo Pascal translator.

The assembler I used for this project is Eric Isaacson's A86 assembler, a marvelous Shareware product that is much faster than the standard Microsoft macro assembler. Although A86 supports some very interesting and useful label conventions, I have endeavored to keep the assembler code for this project completely compatible with the macro assembler. Speaking of marvelous Shareware, I also used Sammy Mitchell's fast and powerful QEDIT programmer's editor.

A powerful and useful feature available in the Marshal system is the separately compiled, Modula 2 style definition module that in turn supports a separately compiled implementation module. The definition module contains standard Pascal constant, type, and variable declarations that are global to any main program module importing it. It also contains declarations of procedures and/or functions that are defined in the implementation module. The implementation module must contain coded definitions of any procedures and/or functions declared in the definition module. It may also contain procedure and/or function definitions not declared in the definition module, but these will be local to the implementation module and not available to the main program. One nice feature that Marshal included in the implementation module is the ability to execute a body of initialization code. Notice that there is a BEGIN END statement pair at the end of SHELTOOL.PAS that includes several lines of code. This code is executed automatically at runtime. In this case, assembly language primitives are called that determine the type of video monitor in use and initialize code segment variables.

Once the definition and implementation modules have been created and compiled, they may be imported into the main Pascal program with the Marshal **import** statement. For this project, I developed and tested a group of routines that I was confident wouldn't require alteration during main program development and placed them in the imported module. That way I could continually change and re-compile the main module without including imported code in the

PRODUCED 2003 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

compile/link process. The Marshal linker handled the task of placing all the object modules into the final executable program.

The listings accompanying this article are organized as follows:

Listing 1: SHELLASM.ASM
Listing 2: SHELTOOL.DEF
Listing 3: SHELTOOL.PAS
Listing 4: SHELL.PAS

Listing 1 contains the assembler code primitives that support this project. Listings 2 and 3 are the Marshal Pascal definition and implementation module source code listings, and Listing 4 contains the main program code.

Two Primary Functions

One of the more important functions of a DOS shell is to provide the user with the capability to load and execute programs. This may be effected through the presentation of an executable program menu or through a "point and shoot" style of interface or both. But no matter how this feature is interfaced, there are two primary functions to implement in order to successfully run a program from the shell.

The job of the first of these functions is to ensure that enough available memory exists in which to execute the program. We can't control the user's total system memory, but we can control how much memory is allocated to the shell itself. When DOS loads and executes a program, it allocates all available memory to the program by default unless the program uses one of the DOS memory allocation functions when it takes control. Our shell has an assembly code function named SET_MEM that does just that.

SET_MEM (see Listing 1) calculates the amount of memory (in 16-byte paragraphs) needed by the shell and then invokes DOS function 4Ah to re-allocate memory to the calculated amount. Since the shell will be compiled and linked using Marshal Pascal's small memory model, the calculations are relatively simple and straightforward. The size of the shell code segment is computed by subtracting the code segment value from the data segment value and then rounding up one paragraph. In Marshal's small memory model, the heap, the stack, and all data are kept in the same segment, so it is necessary only to convert the stack pointer value to paragraphs and then add it to the code segment size in order to calculate total memory required. If an error occurs, SET_MEM returns a-1; otherwise, it returns the number of paragraphs re-allocated to the shell program.

The second function needed to permit execution of a program from the shell is the execute function itself. In this case we have an assembly code primitive that performs the actual task of calling DOS to execute the desired program. This primitive, called EXEC (see Listing 1), is called by a Pascal procedure named EXECUTE (see Listing 2). EXECUTE performs some necessary housekeeping and string manipulation prior to calling EXEC. EXECUTE's main task is to set up the command string passed to it from the main program so that EXEC can pass it to DOS. EXECUTE's housekeeping tasks include saving the current screen, clearing the screen, and restoring the cursor before calling EXEC. When control is returned, EXECUTE must restore the screen and hide the cursor. It also pauses for a user keystroke entry upon completion of the executed program.

The heart of the EXEC primitive is a call to DOS interrupt 2Eh. This is an undocumented call operative in all versions 2.x and greater of DOS. Much has been written in this magazine about the problems associated with using DOS function call 4Bh to load and execute a program. Using interrupt 2Eh is much cleaner and faster and requires far less housekeeping to execute. Moreover, the interrupt 2E call can be used with any command string that could be executed from the DOS level prompt. This means that you can even invoke DOS redirection and piping via this interrupt. And the required registers are relatively simple to set up for the call to this interrupt. Note that the procedure EXECUTE adjusts the command string so that the last position in the string array contains a carriage return and the first position contains a value equal to the length of the string. Also note that the original string may not be longer than 126 characters.

When EXEC gets control, it saves some registers on the stack and then saves the active segment registers, stack pointer, and base pointer in code segment variables. This must be done because interrupt 2Eh destroys all registers, including SS and SP. After this minor bit of housekeeping, the ES register is set to our DS register and SI is set to point to the beginning of the command string. That's all there is to it. When interrupt 2Eh is invoked, it will execute our command string just as if we had typed in the command from the DOS level. Upon return from the interrupt, we restore our segment registers, stack pointer, base pointer, and general registers and return to the caller. The only drawback I see in using interrupt 2Eh is that there is no provision for return of an error level code as there is when using function 4Bh.

The Main Program

Now that we have seen how two of the key primitives work, let's turn our attention to the main program and see how this beast is put together. The first call in the main body is to SET_MEM. The variable PARAGRAPHS might be used to call a routine that checked to see if the host machine has enough system memory. The next call invokes DOS function 0Eh to retrieve the last logical drive that DOS recognizes and then builds a string containing the letters of all legal system drives. This will be used later for a change drive routine.

The next three main body calls clear the screen, save the current cursor size, and then hide the cursor. A call is then made to a procedure that retrieves the default directory name. The procedure GETDIR works very much as the Turbo Pascal intrinsic of the same name does. It returns a string containing not only the current directory path name but the current drive as well. Next, a call to a Pascal initialization procedure is made where the directory boxes and an information box are drawn and the current active directory name and the root directory strings are derived. The next four statements assign values to global array elements that play an important role in exploiting a technique that is the heart and soul of this shell program.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

After initialization is complete, the main body enters a short but very powerful loop that implements the technique I just mentioned. This shell is designed to support as many directory windows as the programmer wants. Each directory window is a port through which the files in a directory may be viewed and manipulated. The example presented here employs only two such windows, each capable of displaying eight lines of file information, but it is easy to modify the shell to employ as many windows as the screen will hold—or the programmer may elect to implement pop-up overlapping windows, in which case any number of windows could be used.

The workhorse of the main body loop is a procedure named VIEW_DIR. VIEW_DIR accomplishes the majority of user interface tasks necessary to view and manipulate files in a specified directory. VIEW_DIR is called with three parameters: an array to hold the individual file data, an integer indicating which window is to be active, and a Boolean flag. When VIEW_DIR gets control, it uses the value parameter **win** to set local variables that control the start and end lines of the window display. The DOS directory name that the window is currently serving is set, and a **chdir** call is made to ensure that the proper directory is now the active directory.

It is at this point that the Boolean variable FLAG comes into play—and where many of you will probably register varying degrees of shock in response to finding a GOTO statement. As an adherent of modular, structured Pascal programming, I generally do not encourage the use of GOTO statements. The GOTO, however, is an intrinsic language feature, and this is an application that can benefit mightily from its use, as we shall see. If FLAG is not set, there is a jump, via the GOTO statement, down to the beginning of a REPEAT UNTIL loop that is contained within a Marshal LOOP structure. (By the way, the fact that this technique works so well is evidence of the robustness of the code generated by the Marshal compiler.) The reason for the jump is so that the user can toggle back and forth between windows without re-initialization of the windows occurring at each entry to VIEW_DIR. By our employment of this technique, the file data array and the cursor location are not disturbed by changing windows. You will notice that the code skipped by the GOTO statement is all initialization code.

Other GOTO statements are used later in the code, but before discussing them I want to go through the statements that lead into the inner REPEAT loop. The SCROLL_IT procedure is used to clear the inside of the directory window box. This is a handy procedure that will be used later to scroll file names up and down in the box. Next, all the files in a window directory are retrieved by the GET_FILES procedure (listed in SHELTOOL.PAS). This is a straightforward routine that employs DOS interrupt 21h, using function calls 4Eh (in FINDFIRST) and 4Fh (in FINDNEXT). The files are placed in an array of records of type FILE_TYPE. Note that the actual file data is transferred directly from the DOS Disk Transfer Area (DTA) to the array element structure via the MOVE procedure. Then the files are sorted using a Shell sort implemented in the Pascal procedure SORT_FILES.

More housekeeping initializes some counters and calculates page counter values. The outer loop of the procedure is then entered, and the file names and associated data are displayed up to eight lines at a time within the window. A function called SHOW is employed to concatenate and format a string containing the file name, time, date, and file size. SHOW calls a function named CONVERT to reformat the file name so that the name and extension are separated by at least one blank space. This adds a degree of readability to the list of file names. Note that using long integers here is necessary to support file sizes above 32,767. The only alternative to long integers is to use single precision (4-byte) reals, but the Marshal floating point module in the runtime library adds about 4K to the code size and slows things down a bit. Marshal does not yet provide an intrinsic long integer type, but they do include long integer support in the form of a separate compilation module called LONGINT. Steve Marshal has promised me that an intrinsic long integer type will be included in the next major release of the compiler.

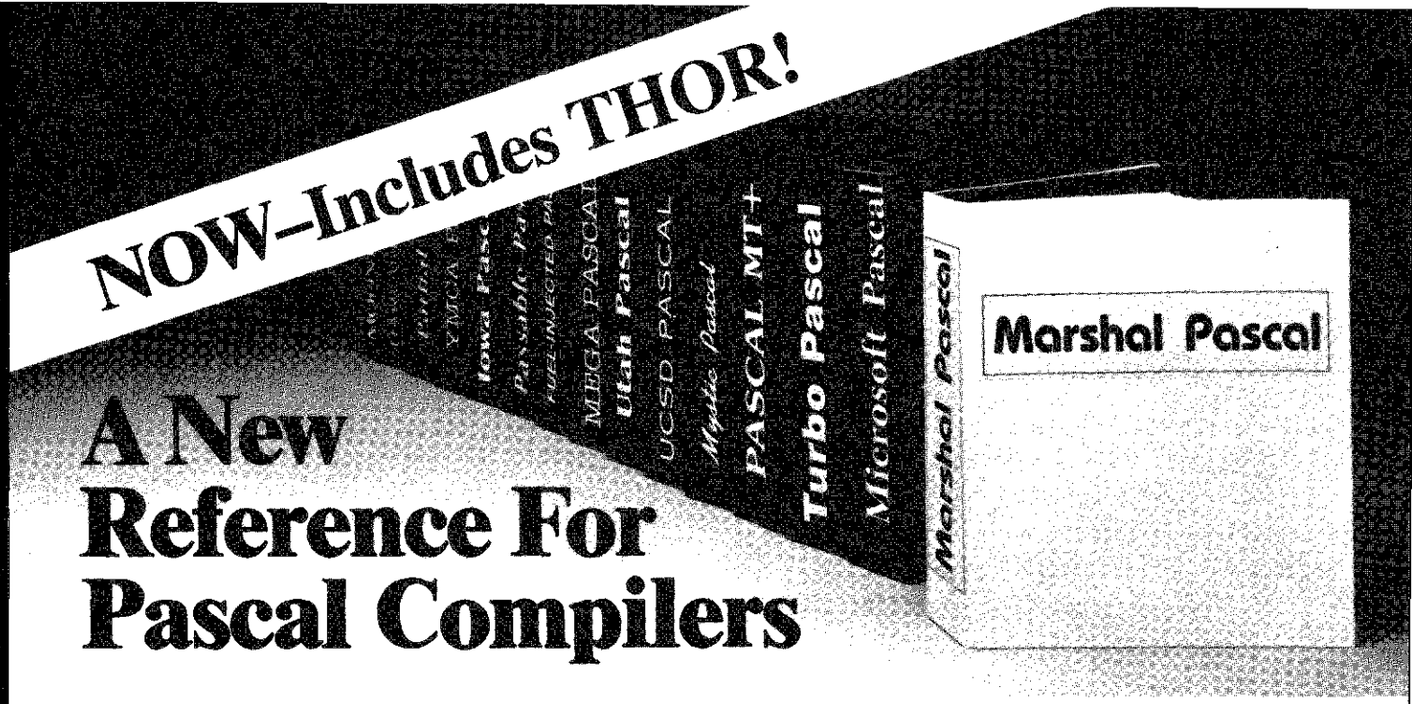
When eight lines have been written or the end of the file array is reached, the procedure enters an inner REPEAT loop. Activity in this loop is controlled by a very effective Boolean function named KEYCODE. KEYCODE returns a false if no key is available in the BIOS keyboard buffer and a true if a key is available. The function returns three integer values—keyboard status, ascii code, and scan code. Having access to these three values any time a key is pressed gives the programmer complete control over interpreting the keyboard. The combination of the scan code and ascii code values is used here to very effectively control loop activity. These activities are

Key combination	Action
Control-Home	Change to root directory.
Control-End	Change to default directory.
Control-PgUp	Change to parent directory.
Return	Execute program or change to indicated directory.
Tab	Switch to other directory window.
C	Copy indicated file to other window directory.
D	Switch to select drive window.
Esc	Terminate the program.
Up/Down arrow	Move cursor bar.
PgUp/PgDn	Page through files.
Del	Delete a file.
Ins	Rename a file.
Home	Show first page of files.
End	Show last page of files.

The REPEAT loop is terminated normally when the user presses the Home, End, PgUp, or PgDn key. Pressing the Esc key calls the TERMINATE procedure, halting the program. The REPEAT loop is broken abnormally by employing a GOTO statement in conjunction with those commands that require re-initialization; for example, pressing the Return key with the cursor positioned over a subdirectory name causes a change of directory to occur. It is then necessary to re-initialize the file array and other supporting variables without terminating the procedure. The GOTO is a very efficient means of accomplishing that. Note also that the Boolean flag is reset to true so that the initialization code will be executed. This scheme is what permits the shell to use just one control procedure for any number of directory windows. Pressing the Tab key toggles back and forth between directory windows so that display, including cursor position, is undisturbed.

A couple of notes about the record structure for each file are in order. First, observe that I have tacked on a field named DESIG at the very end of the record. This field can be used to identify a file as tagged in support of batch move, copy, and delete operations; for example, the plus (+) key could be used to tag files and each time the key were pressed, a unique

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



A New Reference For Pascal Compilers

	Ackerman		Sieve		I/O		Gauss-Seidel		Floating Point	
Marshal Pascal	**11.9**	**3.5K**	**4.8**	**2.4K**	**1.8**	**5.1K**	**4.9**	**11.5K**	**10.5**	**3.2K**
IBM Pascal	12.4	34.7K	11.7	27K	2.5	24.5K	7.6	34.6K	10.5	21K
Turbo Pascal	22.7	11.6K	14.2	11.5K	2.2	12.5K	4.7	13.5K	28	11.4K
Oregon Pascal-2	18.2	13.9K	7.2	11.7K	2.5	22K	11.8	37.6K	139.7	26.8K
Microsoft C 4.0	15.9	9.3K	5.8	6.5K	1.9	8.9K	6.0	23.6K	33	19.6K
	Sec.	Code Size	Sec.	Code Size	Min.	Code Size	Sec.	Code Size	Sec.	Code Size

Unparalleled Speed and Power

Marshal Pascal™ is the most highly code-optimized Pascal compiler for PCs. Period. In fact, Marshal Pascal produces code so fast and compact that even the most efficient C compilers fall behind in performance. Also, Marshal Pascal is an ISO implementation, thus offering portability to other computer environments. You may address as much memory as your operating system allows and a variety of memory models are supported. Among the useful extensions included are: separate compilation of modules (both Modula-2 and Pascal forms), structured constants and structured function values, variable-length string types and procedural parameters.

Turbo Pascal® Translator/Optimizer

Our translator/optimizer shrinks Turbo Pascal software enormously. You can watch your present Turbo programs run in a fraction of their former time and code space!

8087-80287 Support

Marshal Pascal supports the Intel 8087-80287® math processors *inline*. If you don't have the math chip, then '87-287 code simulation is a provided option.

THOR Programming Environment

John Newlin's acclaimed programming environment THOR, along with the professional, windowing text editor THOR-EDIT, are now provided FREE with every copy of Marshal Pascal. Also included at no additional charge are interfaces to **Brief, Edix, Vedit** and other programming editors.

Third Party Support

An ever growing number of third party software companies now provide programming and graphics tools for Marshal Pascal. Among the more prominent firms now providing support for our compiler: **Metagraphics, Softcraft, The Software Bottling Company,** and **Magus,** just to name a few. For an up-to-date list of these companies, please call us for a FREE list and brochure.

The Price? $189, *includes everything.*

Supports PC-DOS®, MS-DOS®, CP/M-86®, Concurrent DOS®, and *soon* Xenix 286/386®!

To order your copy of **Marshal Pascal,** call: **(415) 930-9039**

Number **140** on the Reader Service Card

Marshal Language Systems

1670 Terrace Rd., Walnut Creek, CA 94596

Marshal Pascal is a TM of Marshal Language Systems. Pascal MT+, CP/M-86 and Concurrent DOS are ® of Digital Research, Inc. IBM Pascal, PC-DOS are ® of IBM Corp. Turbo Pascal is a ® of Borland International, Inc. Microsoft TM, Microsoft C, MS-DOS and Xenix are ® of Microsoft Corp. Mystic Pascal is a ® of Mystic Canyon Software. Intel 8087/80287 is a TM of Intel Corp. UCSD Pascal is a TM of Pecan Software Systems. Utah Pascal is a TM of Ellis Computing, Inc. Pascal-2 is a TM of Oregon Software.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Marshal Pascal to Turbo Pascal

Readers who want to convert the Marshal Pascal and supporting assembly source code to code compatible with the Turbo Pascal environment should observe the following steps:

1. Convert the Marshal Pascal code into Turbo Pascal code. This will entail either integrating the code contained in SHELTOOL.PAS and SHELTOOL.DEF into SHELL.PAS or converting SHELTOOL.PAS and SHELTOOL.DEF into an include file readable by the Turbo compiler. Note that the Marshal Pascal intrinsic construct, LOOP, is not supported by the Turbo compiler and will have to be converted into a REPEAT or WHILE loop. Some of the intrinsic Pascal extensions in Turbo Pascal, such as CLRSCR and GOTOXY, will eliminate the need for those procedures in the supporting assembly code.

2. Convert the assembly source code contained in SHELLASM.ASM into a form usable by the Turbo compiler. Note that the Marshal Pascal small code model passes only an offset value in those cases where a VAR parameter is declared while the Turbo compiler passes both the segment and offset values. Another important difference occurs in the return of function results. Marshal passes function results, or pointers thereto, back to the caller via the stack. Turbo passes integer results via the AX register. Note also that Marshal does not include the function result in the byte value declared following the RET statement (for cleaning up the stack) whereas Turbo does.

3. Assemble the converted assembly code into a .COM file (for reference as TURBO external procedures) or into Turbo INLINE statements. If the assembled .COM file is the option you choose, it is recommended that you set up a jump table at the beginning of the source file. The time and labor involved assembling the converted assembly code into Turbo INLINE procedures can be greatly mitigated through the use of an inline assembler such as David Baldwin's outstanding INLINE.COM. ♦

value would be inserted into this field. Later, a loop could examine all file records and perform some operation on the ones marked for action. The second note concerns the size of the record. Both code and space could be saved by omitting the file size fields from the record. This would also reduce the width of the window required and thus permit more on-screen windows. Finally, I want to point out the use of the Marshal compiler directive $U, which effects selective record alignment. Issuing a $U0 directive causes the compiler and linker to byte align the record. A $U1 causes word alignment. Most compilers give you the switch, but it is global and not selective. Without the capability to byte align record structures, a programmer would be unable to access various system structures using Pascal records as templates.

A few words about some supporting procedures are now in order. I am a firm believer in writing directly to the video display RAM. Using the BIOS is just too slow for me. The procedure that does all the display work is FX. It in turn depends upon an assembly language primitive named SCREENWRITE. FX is called with parameters that define the length of the cursor bar, the video attribute of the bar, the column and row (x,y) location of where to begin writing the string, the string to be displayed, and the string's video attribute. Other video routines used in the shell are CLS, SAVE_CURSOR, HIDE_CURSOR, and REST_CURSOR. CLS clears the screen using the parametric attribute value. The others avoid a sloppy display by providing control over the BIOS cursor.

The COPYFILE procedure is not the way I would implement a file copy capability in a commercial product. It is included primarily as a demonstration of using DOS to perform a task via the EXEC procedure. Also, the copy routine is set up to copy from the active directory to the inactive one. This works fine with only two directory windows, but if you add one or more windows, the routine would need to be modified to permit the user to designate the destination directory.

The intrinsic Marshal Pascal procedure PURGE is used to delete a designated file. Renaming a file takes more code since the user must be prompted to enter a new file name. The function USER_ENTRY accomplishes this task. By the way, this is the only place in the shell program where the upper box is used. I included the upper box as an area on the screen where information and error messages can be displayed as well as an area in which to prompt for and accept user entries. The cursor is displayed for the entry, the entry is accomplished, and then the cursor is hidden and the box is cleared. The file is renamed using the Marshal RENAME procedure. Note that Marshal requires opening the file by doing a RESET on it before renaming it. Also, there is no error checking to determine if the file was actually renamed by DOS. You may wish to write your own assembly language primitive that supports error checking and does not require a RESET.

Procedure DRIVE_MENU is the means by which the user can change disk drives. Pressing the D key from either window activates the drive select window. A list of system drives and a bar cursor allow the user to select the desired drive and press Return to effect the drive change. The DOS default directory of the new drive is then active in the window from which the user activated the change drive menu.

Those procedures or functions I have not specifically discussed are straightforward enough that their contribution to the shell program should be evident from studying the Pascal source code. This is also true of the assembly language primitives, many of which I have not discussed due to time and space limitations. All of the assembly code is commented, so there shouldn't be any mystery as to how they work.

As presented here, the shell program lacks many of the functions necessary to make it a full-featured DOS adjunct. Some of the things I would add, for example, include batch operations (move, copy, and delete), a directory listing printout, a file search capability, and disk technical data. But the program is a solid framework for a customized shell to which the programmer can easily add the functions he or she desires. ♦

John J. Newlin is a retired Naval Aviator who currently works for Hughes Aircraft Co. as a systems engineer. He holds degrees in mathematics and computer science and has programmed extensively in Assembly, Pascal, and Modula 2.

Mr. Newlin is the author of SCOUT, a memory resident disk/file/directory manager (reviewed in PC Magazine, *February 24, 1987), and Arcmaster, a "point and shoot" style interface between the user and ARC and de-ARC programs.*

Readers who wish to contact Mr. Newlin may write to him at the following address:

John J. Newlin
4060-228 Rosenda Court
San Diego, CA 92122

Code follows.

PRODUCED 2023 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

★ NOW AVAILABLE FOR C ★ QUICKBASIC ★
★ IBM BASIC/BASICA ★ TURBO PASCAL ★

The Aspen Systems Subroutine Editor (**ASE**) you can call from your programs. Design your own screen layouts - - update in several windows simultaneously. **ASE** is:

- ★ **TRANSPORTABLE** - can be configured for any keyboard and almost any computer or language under MSDOS.
- ★ **VERSATILE** - customize input for any application.
- ★ **COMPLETE** - with full screen edit capabilities and a wide variety of automatic conversions.
- ★ **EASY TO USE** - data and screen layouts described in a single map.
- ★ **FULLY DOCUMENTED** - including examples in BASIC, PASCAL, FORTRAN, C, and even COBOL.
- ★ **AFFORDABLE-**

ASE is still only..... **$99**
DEMO (See what ASE can do for you!)..... **$ 3**

The Aspen Systems Subroutine Package (**ASP**) - the companion package for **ASE** - with many functions unavailable or difficult to perform in high level languages. Like **ASE, ASP** is:

TRANSPORTABLE ★ FULLY DOCUMENTED
EFFICIENT ★ AFFORDABLE

ASP includes 150+ subroutines: conversions; sorts; string, bit, date/time functions; decimal arithmetic, and **MORE**.

ASP is still only..... **$130**

ASE and **ASP** include support and demo programs.

Prices are PPD (continental USA).
Colorado Residents add 3%.

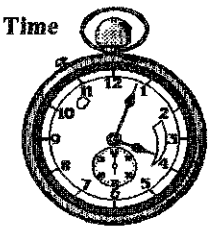
P. O. Box 1163
Grand Junction, CO 81502
(303) 245-3262
VISA/MasterCard accepted

ASPEN SYSTEMS

CP/M and MS-DOS are trademarks of Digital Research and Microsoft, respectively.

Number **86** on the Reader Service Card

Along With Your Computer, Your Time is the Most Important Thing You Own. . .So Why Waste It?



Quilt Programmer Productivity Tools will help you manage your software projects and get control of your time!

SRMS™
Software Revision Management System

- Full Featured Revision Control System
- All Versions stored in a Single ASCII File
- Support for Unlimited Libraries
- Support for all programming languages
- Allows you to use your current compilers and editors without conflict
- Windowing Shell interface to simplify use
- MERGE facility to consolidate different development paths easily, while pointing out conflicting areas
- Full audit trail tracking and reporting on all library components
- Handles big programming projects easily
- Full DOS Pathname and Environment Variable Support
- Requires DOS 2.1+, 224 K, F/H Disk

SRMS Version 3.0.....$185

QMAKE™
Intelligent Program Generation Utility

- Controls the rebuilding of even the most complex systems
- Relieves the developer of remembering which modules need to be rebuilt based on recent changes, how to rebuild them, and in what order to rebuild them
- Works with most compilers, assemblers, and linkers
- Supports full macro definitions, UNIX make-file compatability, recursive invocations, and command line parameters
- Interfaces completely with SRMS, providing you with a complete set of productivity tools to handle any size project
- Requires DOS 2.1+, 128K F/H Disk

QMAKE Version 1.2.....$99

SRMS + QMAKE $250

NEW ! TXT Tools NEW !

QSE - Quilt Text Stream Editor
QSRCH - Quilt File Search Utility (Like UNIX GREP)
QDIFF - Quilt Windowing File Difference Utility

TXTTOOLS Version 1.0..... $85

QUILT COMPUTING

7048 Stratford Road
Woodbury, MN 55125
(612) 739-4650

Volume Discounts and Dealer Inquiries Welcome

Number **46** on the Reader Service Card

WINDOWS—MENUS—DATA ENTRY—SCREENS

HI-SCREEN XL™

If you program, you must try HI-SCREEN *XL* !

HI-SCREEN *XL* is the only tool that gives you all of the following (and much more) whatever programming language you use:

- Unlimited number of screens & windows
- Overlapping windows for menus, data entry, help screens
- Pull-down, pop-up and Lotus-style menus
- On-line help management
- Field checking (for type, range, format,...)
- Field by field and/or full screen input modes
- Scrolling capability
- Full-featured WYSIWYG screen editor
- Redefinable function keys
- Data entry test mode under the editor
- True ease of use, and
- Ease of maintenance: modify screens without recompiling your program.

Includes the HI-SCREEN *XL* Toolbox:

- Capture existing screens from other applications
- Trace compiled programs
- Print screens with clear data field description
- Use screens/windows/menus from DOS and Batch files
- Create titles & logos with predesigned font screens
- ...

Softway, Inc.
PC/Soft Product Line
500 Sutter St., Suite 222
San Francisco, CA 94102

$149 - Risk Free
Visa, M/C welcome
Call (800) 338-2852
in CA (415) 397-4666

For PASCAL, C, dBASE, BASIC, COBOL, FORTRAN

Number **110** on the Reader Service Card

DISK FORMAT CONVERSION

XenoCopy-PC

PC-DOS program lets your PC

$79.95 + $5.00 S/H
Sales Tax if CA.
VISA MasterCard
UPS COD

READ / WRITE /
FORMAT / DUPLICATE
Disks from over 300 other micros

Upgrades available from all previous versions for only $25.00 Call for Authorization

To Order Contact:

XENOSOFT™

1454 Sixth Street, Berkeley, CA 94710
(415) 525-3113

this was on purpose.

Software $ is at work

Number **123** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Code from "The Shell Game" by John Newlin

1

```
;Copyright 1987, John J. Newlin

PUBLIC setxy,upcase,cga_retrace,fillstr,screenwrite,init_screen,savebox
PUBLIC restbox,shiftl,shiftr,hi,lo,move,chdir,findfirst,findnext,cls
PUBLIC strng,video_mode,set_mem,getdir,msdos,set_dta,scroll,keycode
PUBLIC exec,hide_cursor,rest_cursor,save_cursor,longstr,addlong

toolcode    segment word PUBLIC 'code'
            assume cs:toolcode

;;;;;;;;;;;;;;;
;changes lower case char to upper case
;function upcase(ch : char) : char
upcase      proc near
            push bp
            mov bp,sp                       ;save BP
            mov ax,[bp+4]                   ;get the char
            cmp al,97                       ;is it 'a' or above?
            jb u_exit                       ;no, do nothing
            cmp al,122                      ;is it 'z' or below?
            ja u_exit                       ;no, do nothing
            sub al,32                       ;change to upper case
u_exit:     mov [bp+6],ax                   ;return as function result
            pop bp
            ret 2
upcase      endp

;;;;;;;;;;;;;;;
;uses undocumented interrupt 2E to execute a program
;procedure exec(var name : string);
exec            proc near
                push bp                     ;save bp
                mov bp,sp                   ;bp points to the stack
                push ax                     ;save
                push bx                     ; all
                push cx                     ;  other
                push dx                     ;   regs
                push di
                push si
                jmp past                    ;jump over storage area
mls_ds  dw 0                                ;stores our DS reg
mls_es  dw 0                                ;stores our ES reg
mls_ss  dw 0                                ;stores our SS reg
mls_sp  dw 0                                ;stores our SP reg
mls_bp  dw 0                                ;stores our BP reg
past:
                mov cs:mls_ss,ss            ;save SS reg
                mov cs:mls_sp,sp            ;save SP reg
                mov cs:mls_es,es            ;save ES reg
                mov cs:mls_ds,ds            ;save DS reg
                mov cs:mls_bp,bp            ;save BP reg
                push ds
                pop es                      ;set ES = DS
                mov si,[bp+4]               ;si points to command string
                int 2Eh                     ;call DOS to execute
                mov ax,cs:mls_ss            ;retrieve SS reg
                mov ss,ax                   ;reset it
                mov sp,cs:mls_sp            ;retrieve SP reg
                mov ax,cs:mls_ds            ;retrieve DS reg
                mov ds,ax                   ;reset it
                mov ax,cs:mls_es            ;retrieve ES reg
                mov es,ax                   ;reset it
                mov bp,cs:mls_bp            ;retrieve BP reg
                pop si                      ;restore
                pop di                      ; all
                pop dx                      ;  other
                pop cx                      ;   regs
```

2

```
                pop bx
                pop ax
                pop bp
                ret 2
exec            endp

;;;;;;;;;;;;;
;returns keyboard status byte, ascii code, and scan code
;function keycode(var status,ascii,scan : integer) : boolean;
keycode         proc near
                push bp
                mov bp,sp                   ;BP points to stack
                push bx                     ;Marshal says save BX
                push si                     ;and also SI
                mov ah,1                    ;Int 16h to check buffer
                int 16h                     ;call BIOS
                mov word ptr [bp+10],0      ;function returns a false
                jz _no_key                  ;no key avail, so depart
                mov ah,0                    ;key is avail, so get it
                int 16h                     ;call BIOS
                push ax                     ;save result
                mov ah,2                    ;to get status byte
                int 16h                     ;call BIOS
                mov bx,[bp+8]               ;get address of status
                mov ah,0                    ;zero the MSB
                mov [bx],ax                 ;set status
                pop ax                      ;restore AX
                mov bx,[bp+6]               ;get address of ascii
                mov si,[bp+4]               ;get address of scan
                xor dx,dx                   ;zero DX
                mov dl,ah                   ;put scan value in DX
                mov [si],dx                 ;move it into scan
                mov dl,al                   ;move ascii code into DL
                mov [bx],dx                 ;and into ascii
                mov word ptr[bp+10],1       ;function returns true
_no_key:        pop si
                pop bx
                pop bp
                ret 6
keycode         endp

;;;;;;;;;;;;;;;
;scrolls the specified area of the screen up/down number of lines
;normalize to screen dimensions of (1,1) to (25,80)
;procedure scroll(ulx,uly,lrx,lry,lines,attr,dir : integer); external;
scroll          proc near
                push bp
                mov bp,sp                   ;BP points to stack
                push bx                     ;must preserve BX
                mov bh,byte ptr 6[bp]       ;get desired attribute
                mov ch,byte ptr 14[bp]      ;upper left y coord
                dec ch                      ;normalize
                mov cl,byte ptr 16[bp]      ;upper left x coord
                dec cl                      ;normalize
                mov dh,byte ptr 10[bp]      ;lower right y coord
                dec dh                      ;normalize
                mov dl,byte ptr 12[bp]      ;lower right x coord
                dec dl                      ;normalize
                mov al,byte ptr 08[bp]      ;no. of lines to scroll
                cmp byte ptr 4[bp],0        ;scroll up?
                je scroll_up                ;yes
                mov ah,7                    ;no, scroll down
                jmp short scroll_it         ;do it
scroll_up:      mov ah,6                    ;scroll up
scroll_it:      int 10h                     ;call BIOS
                pop bx
                pop bp
```


PRODUCED 2003 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
                ret 14
scroll          endp

;;;;;;;;;;;;;
;set the Disk Transfer Area to desired structure
;procedure set_dta(var buffer : buff_type);
set_dta         proc near
                push bp
                mov bp,sp               ;BP points to stack
                mov dx,[bp+4]           ;get address of buffer
                mov ah,1Ah              ;function to set DTA
                int 21h                 ;DOS will do it
                pop bp
                ret 2
set_dta         endp

;;;;;;;;;;;;;;
;call DOS interrupt 21h with registers set as desired
;procedure msdos(var regs : regtype)
msdos           proc near
                push bp
                mov bp,sp               ;BP points to stack
                push ax                 ;save
                push bx                 ; all
                push cx                 ; regs
                push dx                 ;
                push di                 ;
                push si                 ;
                push ds                 ;
                push es                 ;
                jmp short pst           ;leap over code seg data
saveit dw 0 pst:
                mov bx,[bp+4]           ;BX points to regs structure
                mov ax,[bx+0]           ;set AX
                mov cx,[bx+4]           ;set CX
                mov dx,[bx+6]           ;set DX
                mov di,[bx+8]           ;set DI
                mov si,[bx+10]          ;set SI
                mov cs:saveit,ds        ;save our DS reg
                push [bx+14]            ;push ES value in structure
                pop es                  ;pop it into ES
                push [bx+2]             ;push BX value in structure
                push [bx+12]            ;push DS value in structure
                pop ds                  ;set DS
                pop bx                  ;set BX
                push bp                 ;save our BP during call
                int 21h                 ;call DOS
                pop bp                  ;restore our BP
                push ds                 ;save returned DS
                push bx                 ;save returned BX
                mov bx,cs:saveit        ;get our DS reg
                mov ds,bx               ;set our DS
                mov bx,[bp+4]           ;get pointer to structure
                pop [bx+2]              ;set structure BX
                pop [bx+12]             ;set structure DS
                mov [bx+0],ax           ;set structure AX
                mov [bx+4],cx           ;set structure CX
                mov [bx+6],dx           ;set structure DX
                mov [bx+8],di           ;set structure DI
                mov [bx+10],si          ;set structure SI
                push es                 ;push returned ES
                pop [bx+14]             ;pop it into structure ES
                pushf                   ;push returned flags
                pop [bx+16]             ;pop into structure flags
                pop es                  ;restore
                pop ds                  ; all
                pop si                  ;  regs
```



```
                pop di                  ;
                pop dx                  ;
                pop cx                  ;
                pop bx                  ;
                pop ax                  ;
                pop bp                  ;
                ret 2
msdos           endp

;;;;;;;;;;;;;
;returns active drive and directory in string
;drive code 0 = A, 1 = B, etc.
;procedure getdir(var dirname : string);
getdir          proc near
                push bp
                mov bp,sp               ;BP points to stack
                push si                 ;must preserve SI
                mov si,[bp+4]           ;SI points to dirname
                mov ah,19h              ;set up to get drive desig
                int 21h                 ;call DOS
                add al,65               ;make it a char
                mov byte ptr[si],al     ;put in string
                inc si                  ;point to next char
                mov al,58               ;58 = ':'
                mov byte ptr[si],al     ;put in string
                inc si                  ;point to next char
                mov byte ptr[si],'\'    ;put in backslash
                inc si                  ;point to next char
                mov ah,47h              ;set up to get default dir
                int 21h                 ;call DOS
                pop si
                pop bp
                ret 2
getdir          endp

;;;;;;;;;;;;;;
;set program memory to only what is required by code, data, and stack
;function set_mem : integer;
set_mem         proc near
                push bp                 ;save BP
                mov bp,sp               ;BP points to stack
                push es                 ;save ES
                push bx                 ;save BX
                mov ax,ds               ;AX = DS
                mov bx,cs               ;BX = CS
                sub ax,bx               ;ds-cs = data seg paragraphs
                inc ax                  ;round up one
                mov bx,ax               ;store in BX
                mov ax,bp               ;AX = stack pointer
                mov cl,4                ;for shifting
                shr ax,cl               ;convert to paragraphs
                add ax,bx               ;add BX
                add ax,32               ;bump by 32 paragraph
                                        ;overhead
                mov bx,ax               ;store in BX
                mov cx,ax               ;save in CX
                mov ax,cs               ;AX = code segment
                sub ax,16               ;AX set to mem control block
                                        ;seg
                mov es,ax               ;ES = mem control block seg
                mov ah,4Ah              ;DOS mod alloc mem function
                int 21h                 ;DOS will do it
                jnc no_mem_err          ;error will set carry flag
                mov cx,-1               ;if error return -1
no_mem_err:     mov [bp+4],cx           ;function result
                pop bx                  ;restore BX
                pop es                  ;restore ES
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
                pop bp                  ;restore BP
                ret
set_mem         endp

;;;;;;;;;;;;;;;
;converts an integer into a Pascal string
;procedure strng(num : integer; var numstr : string)
strng           proc near
                push bp
                mov bp,sp               ;BP points to stack
                push bx                 ;preserve BX
                push di                 ;preserve DI
                push es                 ;preserve ES
                push ds                 ;preserve DS
                mov ax,ds               ;get DS
                mov es,ax               ;make ES = DS
                mov di,[bp+4]           ;DI points to string
                mov dx,[bp+6]           ;AX = integer to convert
                xor bx,bx               ;zero BX
new_char:       mov ax,dx
                xor dx,dx               ;zero DX
                mov cx,10               ;for decimal conversion
                div cx                  ;AX gets result
                xchg ax,dx              ;swap AX and DX
                add al,30h              ;convert to ascii
                push ax                 ;save AX
                inc bx                  ;bump count
                cmp dx,0                ;are we done?
                jnz new_char            ;no, get another digit
str_loop:
                pop ax                  ;restore AX
                stosb                   ;put in string
                dec bx                  ;dec count
                cmp bx,0                ;are we done?
                jne str_loop            ;no, do again
                mov al,0                ;AL = 0
                stosb                   ;end of string byte
                pop ds                  ;restore
                pop es                  ; regs
                pop di                  ;
                pop bx                  ;
                pop bp                  ;
                ret 4
strng           endp

;;;;;;;;;;;;
;clears the entire screen using the specified attribute
;procedure cls(attr : integer);
cls             proc near
                push bp
                mov bp,sp               ;BP points to stack
                push bx                 ;preserve BX
                mov ah,7                ;code to scroll
                mov al,0                ;AL = 0 so all will be
                                        ;blanked
                mov bh,byte ptr[bp+4]   ;attribute for scroll
                mov cx,0                ;CL = 0, CH = 0
                mov dx,184Fh            ;DL = 79, DH = 24
                int 10h                 ;call BIOS
                pop bx                  ;restore BX
                pop bp
                ret 2
cls             endp

;;;;;;;;;;;;;;;;;
```



```
;returns current video mode
;function vide_mode : integer
video_mode      proc near
                push bp
                mov bp,sp               ;BP points to stack
                mov ah,0Fh              ;set up to get mode
                int 10h                 ;call BIOS
                mov ah,0                ;clear upper half
                mov 4[bp],ax            ;return function result
                pop bp
                ret
video_mode      endp


;;;;;;;;;;;;;;;;;;
;if file is found, returns a 0 - returns non-zero otherwise
;function findfirst(var pathname : string; attr : integer) : integer
findfirst       proc near
                push bp
                mov bp,sp               ;BP points to stack
                mov dx,[bp+6]           ;DX points to ASCIIZ string
                mov cx,[bp+4]           ;CX = file attribute
                mov ah,4Eh              ;set up to find first
                int 21h                 ;call DOS
                jc ff_err               ;carry flag set if not found
                mov ax,0                ;was found, so AX = 0
ff_err:         mov [bp+8],ax           ;return function result
                pop bp
                ret 4
findfirst       endp

;;;;;;;;;;;;;;;;;;
;returns 0 if next find is successful, non-zero if not
;function findnext : integer
findnext        proc near
                push bp
                mov bp,sp               ;BP points to stack
                mov ah,4Fh              ;set up for find next
                int 21h                 ;call DOS
                jc fn_err               ;carry flag set if not found
                mov ax,0                ;was found, so AX = 0
fn_err:         mov [bp+4],ax           ;return function result
                pop bp
                ret
findnext        endp

;;;;;;;;;;;;;;;;;;;;;;;
;changes to specified drive/directory
;returns 0 if successful, non-zero if not
;function chdir(var dirname : string) : integer;
chdir           proc near
                push bp
                mov bp,sp               ;BP points to stack
                push si                 ;must preserve SI
                mov si,[bp+4]           ;SI points to dirname
                cmp byte ptr [si+1],':' ;is drive spec present
                jne no_drive            ;if not, skip drive stuff
                mov dl,[si]             ;yes, get drive letter
                cmp dl,65               ;is it below 'A'
                jb no_drive             ;yes, skip
                cmp dl,90               ;is it above 'Z'
                ja lo_case_desig        ;yes, check lower case
                sub dl,65               ;convert letter to byte
                jmp drv_ok              ;continue
lo_case_desig:  cmp dl,97               ;is it below 'a'
                jb no_drive             ;yes, skip
                cmp dl,122              ;is it above 'z'
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
                ja no_drive             ;yes, skip
                sub dl,97               ;convert letter to byte
drv_ok:         mov ah,0Eh              ;change drive
                int 21h                 ;call DOS
                cmp byte ptr [si+2],0   ;are we done?
                je are_done             ;yes
no_drive:       mov dx,si               ;dx points to directory
                mov ah,3Bh              ;set up to change
                int 21h                 ;call DOS
                jc ch_err               ;if err don't zero AX
are_done:       mov ax,0                ;AX = 0
ch_err:         mov [bp+6],ax           ;return function result
                pop si                  ;restore SI
                pop bp
                ret2
chdir           endp

;;;;;;;;;;;;;;;
;returns most significant byte of num
;function hi(num : integer) : integer;
hi              proc near
                push bp
                mov bp,sp               ;BP points to stack
                mov ax,[bp+4]           ;AX = num
                xchg ah,al              ;swap
                mov ah,0                ;clear upper half
                mov [bp+6],ax           ;return function result
                pop bp
                ret 2
hi              endp

;;;;;;;;;;;;;;;;;
;returns least significant byte of num
;functio lo(num : integer) : integer;
lo              proc near
                push bp
                mov bp,sp               ;BP points to stack
                mov ax,[bp+4]           ;AX = num
                mov ah,0                ;clear upper half
                mov [bp+6],ax           ;return function result
                pop bp
                ret 2
lo              endp
;;;;;;;;;;;;;;;;;
;shifts target to the left by quantity bits
;function shiftl(target,bits : integer) : integer;
shiftl          proc near
                push bp
                mov bp,sp               ;BP points to stack
                mov cl,byte ptr [bp+4]  ;CL = bits
                mov ax,[bp+6]           ;AX = target
                shl ax,cl               ;shift left
                mov [bp+8],ax           ;return function result
                pop bp
                ret 4
shiftl          endp

;;;;;;;;;;;;;;;;;
;shifts target to the right by quantity bits
;function shiftr(target,bits : integer) : integer;
shiftr          proc near
                push bp
                mov bp,sp               ;BP points to stack
                mov cl,byte ptr [bp+4]  ;CL = bits
                mov ax,[bp+6]           ;AX = target
                shr ax,cl               ;shift right
                mov [bp+8],ax           ;return function result
```



```
                pop bp
                ret 4
shiftr          endp

;;;;;;;;;;;;;;;;;;
;initializes video ram address and video type variables
;procedure init_screen;
init_screen     proc near
                push bp
                jmp short past_data
vid_addr dw 0B000h
mono_box db 1
past_data:      push ds                 ;save DS
                push bx                 ;save BX
                mov ah,0Fh              ;set up to get mode
                int 10h                 ;call BIOS
                cmp al,7                ;is it mono?
                je mc                   ;yes
                mov ax,0B800h           ;no, set up for CGA
                mov cl,0                ;for mono_box
                jmp short continue      ;and continue
mc:             mov ax,0B000h           ;set up for mono
                mov cl,1                ;for mono_box
continue:       mov cs:vid_addr,ax      ;save video address
                mov cs:mono_box,cl      ;save the flag
                pop bx                  ;restore BX
                pop ds                  ;restore DS
                pop bp
                ret
init_screen     endp

;;;;;;;;;;;;;;;;;;;
;waits for horiz and vertical retrace before writing to video ram
cga_retrace     proc near
retrace:        in al,dx                ;get status
                test al,8               ;check retrace
                jnz horiz               ;ok - chech horiz
                rcr al,1                ;check retrace
                jc retrace              ;check again
horiz:          in al,dx                ;get status
                rcr al,1                ;check retrace
                jnc horiz               ;test again
store_it:        ret
cga_retrace     endp


;;;;;;;;;;;;;;;;;;
;saves the video area defined by row,col, width and depth to buff
;procedure savebox(col,row,width,depth,buff);
savebox         proc near
                push bp
                mov bp,sp               ;BP points to stack
                push ds                 ;save
                push es                 ; important
                push bx                 ;   regs
                push di                 ;    on
                push si                 ;     stack
                push ss
                pop es                  ;ES = SS
                jmp short mc2           ;jump over data and proc

s_width  dw 0
s_offset dw 0
mc2:            mov ax,cs:vid_addr      ;get video address
continue2:      mov ds,ax               ;put in DS
                mov di,[bp+4]           ;DI points to buffer
                mov ax,[bp+10]          ;AX = row
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
                dec ax                  ;normalize
                mov bx,160              ;160 bytes per row
                mul bx                  ;times number of rows
                mov si,ax               ;SI points to video start
                mov bx,[bp+12]          ;BX = col
                dec bx                  ;normalize
                shl bx,1                ;times 2
                mov dx,[bp+8]           ;DX = width
                shl dx,1                ;times 2
                mov cs:s_width,dx       ;save width
                mov cs:s_offset,160     ;save offset
                sub cs:s_offset,dx      ;subtract width
                add si,bx               ;add to SI
                mov cx,[bp+6]           ;CX = depth
                cmp cs:vid_addr,0B000h  ;mono screen?
                mov ax,8[bp]            ;AX = width
                je mono9                ;yes, go to mono routine
again9:         push cx                 ;save CX = depth
                mov cx,[bp+8]           ;CX = width
                mov dx,03DAh            ;CGA port
loop9:          cli                     ;halt interrupts
                call cga_retrace        ;wait for CGA retrace
store9:         movsw                   ;store the char and attribute
                sti                     ;restore interrupts
                loop loop9              ;do again
                pop cx                  ;restore CX = depth
                add si,cs:s_offset      ;add the offset
                loop again9             ;and continue
                jmp y_done              ;until done
mono9:          push cx                 ;save CX = depth
                mov cx,ax               ;CX = AX = width
                rep movsw               ;do it
                add si,cs:s_offset      ;add the offset
                pop cx                  ;restore CX = depth
                loop mono9              ;do again
y_done:         pop si                  ;restore
                pop di                  ; regs
                pop bx                  ;  from
                pop es                  ;    the
                pop ds                  ;     stack
                pop bp
                ret 10
savebox         endp

;;;;;;;;;;;;;;;;;;;
;restores buffer to video area defined by col, row, width, and depth
;procedure restbox(col,row,width,depth,buff : integer)
restbox         proc near
                push bp
                mov bp,sp               ;BP points to stack
                push ds                 ;save
                push es                 ; important
                push bx                 ;  regs
                push di                 ;    on
                push si                 ;     stack
                mov ax,cs:vid_addr      ;AX = video address
                mov es,ax               ;ES = video address
                mov si,[bp+4]           ;SI points to buff
                mov ax,[bp+10]          ;AX = row
                dec ax                  ;normalize
                mov bx,160              ;160 bytes per row
                mul bx                  ;times number of rows
                mov di,ax               ;DI points to area
                mov bx,[bp+12]          ;BX = col
                dec bx                  ;normalize
                shl bx,1                ;times 2
                mov dx,[bp+8]           ;DX = width
```



```
                shl dx,1                ;times 2
                mov cs:s_width,dx       ;store width
                mov cs:s_offset,160     ;store offset
                sub cs:s_offset,dx      ;DX = offset
                add di,bx               ;DI points to area
                mov cx,[bp+6]           ;CX = depth
                cmp cs:mono_box,1       ;mono screen?
                mov ax,[bp+8]           ;AX = video segment
                je mono6                ;go to mono routine
again6:         push cx                 ;save CX = depth
                mov cx,[bp+8]           ;CX = width
                mov dx,03DAh            ;CGA port address
loop6:          cli                     ;turn off interrupts
                call cga_retrace        ;wait for retrace
store6:         movsw                   ;store char and attribute
                sti                     ;interrupts enabled
                loop loop6              ;do again
                pop cx                  ;CX = depth
                add di,cs:s_offset      ;add offset to DI
                loop again6             ;and do another row
                jmp x_done              ;done with CGA
mono6:          push cx                 ;save CX = depth
                mov cx,ax               ;CX = width
                rep movsw               ;move all at once
                add di,cs:s_offset      ;add offset to DI
                pop cx                  ;CX = depth
                loop mono6              ;do another row
x_done:         pop si                  ;restore
                pop di                  ; regs
                pop bx                  ;  from
                pop es                  ;    the
                pop ds                  ;     stack
                pop bp                  ret 10
restbox         endp

;;;;;;;;;;;;;;;;;;;;;;;
;writes a string directly to video ram
; procedure screenwrite(col,row,attr:integer; var str : string);
screenwrite     proc near
                push bp
                mov bp,sp               ;BP points to stack
                push es                 ;save
                push bx                 ; regs
                push di                 ;  on
                push si                 ;   stack
                mov ax,[bp+8]           ;AX = row
                dec ax                  ;normalize
                mov bx,160              ;160 bytes per row
                mul bx                  ;AX = AX * 160
                mov di,ax               ;DI points to video area
                mov bx,[bp+10]          ;BX = col
                dec bx                  ;normalize
                shl bx,1                ;adjust for attr byte
                add di,bx               ;DI = DI + col
                mov si,[bp+4]           ;SI points to string
                mov ah,[bp+6]           ;AH = attribute
                mov dx,cs:vid_addr      ;DX = video segment
                cmp byte ptr cs:mono_box,1  ;mono screen?
                mov es,dx               ;ES = video segment
                je sw_mono              ;yes, go to mono routine
                mov dx,03dah            ;CGA port address
sw_getnext:     lodsb                   ;AL = char
                cmp al,0                ;end of string?
                je sw_exit              ;yes, exit
                mov cx,ax               ;no, save AX in CX
                cli                     ;disable interrupts
                call cga_retrace        ;wait for retrace
```


PRODUCED 2003 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

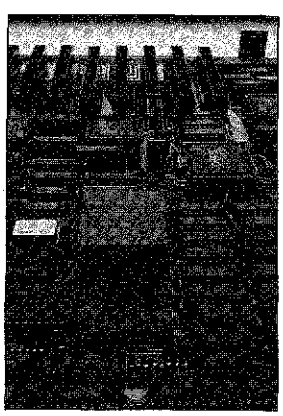
SAVE YOUR PC FROM EARLY RETIREMENT.

GET HAUPPAUGE'S NEW 386 MOTHERBOARD.

386 SPEED–ONLY $1,495

Give your PC a new lease on life! With our industry first 386 MotherBoard, your PC, PC/XT or compatible will revel in speeds equal to the Compaq DeskPRO 386. And faster. Because we've built in 1 Megabyte of high speed RAM and a 387 math coprocessor socket for speeds that will knock you off your rocker.

To keep retirement at bay, our 386 MotherBoard is compatible with the PC/AT (BIOS and I/O) — allowing you to run the new generation of DOS, OS/2. We've also included a 16-bit expansion slot that accommodates the latest I/O expansion card. No accelerator card can give you so much versatility.



With 386 power and true AT software compatibility, your business, desktop publishing and engineering applications will get a boost to boast about! **Technical Features** ▪ 16 MHz 80386 ▪ 1 Megabyte of 100 nsec 4-way interleaved RAM ▪ PC/AT compatible I/O and BIOS for support of OS/2 ▪ Seven 8-bit expansion slots ▪ Two 16-bit expansion slots ▪ One 32-bit RAM expansion slot ▪ Optional 16 MHz 80387 math coprocessor ($695)

Put the power of the 386 into your IBM PC for 1/4 the cost of a 386 computer. And put off your PC's retirement. For more information on our easy-to-install Motherboard, call **1 (800) 443-6284.** In New York, call **(516) 360-3827.**

Hauppauge Computer Works, Inc.
358 Veterans Memorial Highway,
Commack, New York 11725

Hauppauge!

Trademarks: IBM PC/AT ... Compaq Computers

Number 122 on the Reader Service Card.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
sw_store:       mov ax,cx               ;restore AX
                stosw                   ;store word in video ram
                sti                     ;enable interrupts
                jmp sw_getnext          ;get next character
sw_mono:        lodsb                   ;AL = next char
                cmp al,0                ;end of string?
                je sw_exit              ;yes, exit
                stosw                   ;no, store in video ram
                jmp sw_mono             ;get next character
sw_exit:        pop si                  ;restore
                pop di                  ; regs
                pop bx                  ;  from
                pop es                  ;   stack
                pop bp
                ret 8
screenwrite     endp


;;;;;;;;;;;;;;;;;;;;;
;moves bytes from vladdr to v2addr
;procedure move(vladdr,v2addr,bytes : integer);
move            proc near
                push bp
                mov bp,sp               ;BP points to stack
                push es                 ;save
                push di                 ; regs
                push si                 ;  on stack
                mov ax,ds               ;AX = DS
                mov es,ax               ;ES = AX = DS
                mov si,[bp+8]           ;SI points to vladdr
                mov di,[bp+6]           ;DI points to v2addr
                mov cx,[bp+4]           ;CX number of bytes to move
                rep movsb               ;mass move
                pop si                  ;restore
                pop di                  ; regs
                pop es                  ;  from stack
                pop bp
                ret 6
move            endp

;;;;;;;;;;;;;;;;;;;;;
;positions cursor to col, row values
;procedure setxy(col,row : integer)
setxy           proc near
                push bp
                mov bp,sp               ;BP points to stack
                push bx                 ;preserve BX
                mov bh,0                ;page is 0
                mov ah,2                ;for BIOS call to move cursor
                mov dh,[bp+4]           ;DH = row
                dec dh                  ;normalize
                mov dl,[bp+6]           ;DL = col
                dec dl                  ;normalize
                int 10h                 ;BIOS will move it
                pop bx                  ;restore BX
                pop bp
                ret 4
setxy           endp

;;;;;;;;;;;;;;;;;;;;;
;fills designated string with specified character
;procedure fillstr(var target : string; num,ch : integer);
fillstr         proc near
                push bp
                mov bp,sp               ;BP points to stack
                push es                 ;save ES
                mov ax,ds               ;AX = DS
```



```
                mov es,ax               ;ES = AX = DS
                mov di,8[bp]            ;DI points to target string
                mov cx,6[bp]            ;CX = num
                cmp cx,80               ;longer than 80 char?
                jg poof                 ;yes, depart
                mov al,byte ptr 4[bp]   ;AL = ch
                cld                     ;for forward move
                rep stosb               ;store ch in string
                mov al,0                ;AL = 0
                stosb                   ;for end of string byte
poof:           pop es                  ;restore ES
                pop bp
                ret 6
fillstr         endp

;;;;;;;;;;;;;;;;;;;;;
;adds two long integers - return result in total
;procedure addlong(var total,n1,n2 : longint)
addlong         proc near
                push bp
                mov bp,sp
                push bx                 ;save
                push di                 ; regs
                push si                 ;  on stack
                mov si,[bp+6]           ;SI points to n1 low word
                mov di,[bp+4]           ;DI points to n2 low word
                mov bx,[bp+8]           ;BX points to total low word
                mov cx,2                ;CX = 2
                clc                     ;clear carry
add_em:         mov ax,[si]             ;AX = n1
                inc si                  ;bump SI
                inc si                  ;SI now points to n1 hi word
                adc ax,[di]             ;add n1 to n2
                inc di                  ;bump di
                inc di                  ;DI now points to n2 hi word
                mov [bx],ax             ;store in total lo word
                inc bx                  ;bump bx
                inc bx                  ;BX now points total hi word
                loop add_em             ;do again for hi word
                pop si                  ;restore
                pop di                  ; regs
                pop bx                  ;  from stack
                pop bp
                ret 6
addlong         endp


;;;;;;;;;;;;;;;;;;;;;
;converts a long integer into a string;
;procedure longstr( var long : longint; var str : longstr)
longstr         proc near
                push bp
                mov bp,sp               ;BP points to stack
                push ds                 ;save
                push es                 ; regs
                push bx                 ;  on
                push si                 ;   the
                push di                 ;    stack
                jmp over_dat            ;skip over data
temp_buff db 5 dup(0)
lstr db 12 dup(0)
over_dat:       cld                     ;set direction to forward
                mov ax,ds               ;AX = DS
                push ax                 ;save AX, we need DS later
                mov bx,[bp+6]           ;BX points to long low word
                mov dx,[bx+2]           ;DX points to long hi word
                mov bx,[bx]             ;BX = long low word
```


PRODUCED 2008 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Upgrade your technology

The software technology available to programmers of IBMcompatible personal computers is truly amazing. And newer, more powerful development packages appear all the time. But until now, finding out about these important products has been a difficult and time consuming task.

FREE Buyer's Guide. The New 76 page Programmer's Connection Fall 1987 Buyers Guide contains individual descriptions of over 500 titles of programmer's development software by over 150 manufacturers. Each description covers major product features as well as any software or hardware requirements and version numbers. In the box on the right are some examples of the types of descriptions you'll find in our Buyer's Guide.

No Hidden Charges. The low discount prices in our Buyer's Guide are all you pay. We don't charge extra for UPS Ground shipping, credit cards, COD orders, purchase orders, sales tax (except Ohio) or special handling (except for non-Canadian international orders).

Guarantees. We offer FREE 30-day no-risk return guarantees and 30-day evaluation periods on most of our products.

Latest Versions. The products we carry are the latest versions and come with the same manufacturer's technical support as if buying direct.

Large Inventory. We have one of the largest inventories of programmer's development products in the industry. Most orders are shipped within 24 hours.

Noncommissioned Staff. Our courteous salespeople are always ready to help you. And if you aren't sure about your needs, our knowledgeable technical people can give you sound, objective advice.

Experience. We've specialized in development software for IBMcompatible personal computers since 1984 and are experienced at providing a full range of quality products and customer services.

How to Get Your Copy. There are three ways for you to receive your FREE copy of the Programmer's Connection Buyer's Guide: 1) Use the reader service card provided by this journal; 2) Mail us a card or letter with your name and address; or 3) Call one of our convenient toll free telephone numbers.

If you haven't yet received your Programmer's Connection Fall 1987 Buyer's Guide, act now. Upgrading your programming technology could be one of the wisest and most profitable decisions you'll ever make.

CALL TOLL FREE
USA: 800-336-1166
Canada: ... 800-225-1166
Ohio & Alaska (Collect):..216-494-3781
International: 216-494-3781
Telex: 9102406879
Easylink:..... 62806530

Programmer's Connection
7249 Whipple Ave. NW
North Canton, OH 44720

Please turn the page for our latest price list and ordering information.

Blaise
C TOOLS PLUS/5.0
List $129 Ours $99

C TOOLS PLUS/5.0 is a library for Microsoft C that can provide you with a full spectrum of general-purpose utility functions. Included are functions that take advantage of the machine features of IBM-compatible personal computers and DOS and complement the standard compiler library. Almost all functions are written in C using techniques most suitable for good C program design. Some of the areas covered are: extensive string handling; screen handling including support for multiple monitors and the EGA; general utility and keyboard functions; DOS memory management; windows that can be stacked, removed and accept user input; intervention code; and interrupt service routine support for truly flexible resident applications. C TOOLS PLUS/5.0 includes all source code, complete examples and a comprehensive reference manual.

Supports Microsoft C 5.0 and Microsoft QuickC.

ESP
Command Plus
List $80 Ours $69

Command Plus is a powerful, bootable MS-DOS command processor fully compatible with COMMAND.COM. Many new features are included that will help you increase your speed, productivity and ease of programming. These include a history processor that lets you recall and edit previously entered commands using the cursor keys. The alias facility allows the creation of command macros with replaceable parameters. Other features include: regular expressions in filenames; directory and argument stacks; a command line editor; access to environment variables; a MOVE command; and Browse, a full screen file viewer. Command Plus also includes SCRIPT, a batch processor that uses a Pascal-like language that features: integer and string variables; boolean, math and string operators; CALL, FOR/WHILE, GOTO, IF/THEN/ELSE, and SWITCH statements; and display and file access routines. Enhancements to DIR include sort options and file attrib display. COPY options include selection by date/time range, and recursive sub-directory processing.

Requires 48K memory. Version 1.2.

Meridian Software
AdaVantage Compiler 2.0
List $795 Ours $735

The Meridian AdaVantage Compiler is a fully validated implementation of the Ada language. The compiler generates native 8086 code in the Intel standard object format. A linker is provided to create stand-alone executable files for MS-DOS. The Meridian AdaVantage Compiler package includes the compiler, linker, library management tools, support packages, runtime libraries and a configuration tool. In addition to the standard Ada packages, support packages are provided to make integer, floating point and text I/O more convenient to implement. The compiler also provides a pragma INTERFACE used to make calls to subprograms written in 8086 assembly language, Meridian C or Meridian Pascal. A source level debugger will be available in late 1987.

Requires hard disk and 640K memory. Runtime fees apply if more than 99 copies are sold. Version 2.0.

PROGRAMMER'S QUALITY INTEGRITY Service SUPPORT PRICE CONNECTION

The Free Programmer's Connection Fall '87 Buyer's Guide.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

ai - expert systems

	List	Ours
1st-CLASS *by Programs in Motion*	495	399
EXSYS *Development Software by EXSYS*	395	309
EXSYS Runtime System	600	469
LEVEL5 *by Information Builders* ... *New*	685	569
Logic-Line Series *All varieties by Thunderstone*	CALL	CALL

ai - lisp language

Golden Common LISP *by Gold Hill*	495	CALL
Golden Common LISP Developer *by Gold Hill*	1190	CALL
Q'Nial *Various by NIAL Systems*	CALL	CALL
Star Sapphire LISP *Compiler by Sapiens* ... *New*	495	429
TransLISP PLUS *from Solution Systems*	195	125

ai - prolog language

Arity Combination Package	1095	979
Expert System Development Pkg	295	229
File Interchange Toolkit	50	44
PROLOG Compiler & Interpreter	650	569
Screen Design Toolkit	50	44
SQL Development Package	295	229
Arity PROLOG Interpreter	295	229
Arity Standard Prolog	95	77
LPA microPROLOG *All Varieties*	CALL	CALL
MPROLOG Language Primer *LOGICWARE*	50	45
MPROLOG P500 *by LOGICWARE*	495	395
MPROLOG P550 *w/Primer by LOGICWARE*	220	175
Turbo PROLOG *by Borland Intl*	100	64
Turbo PROLOG Toolbox *by Borland Intl*	100	64

ai - smalltalk language

Smalltalk/V	100	84
EGA/VGA Color Option	50	45
Goodies Diskette	50	45
Smalltalk/Comm	50	45

ai - texas instruments

Arborist *Decision Tree Software*	595	519
PC Scheme *Lisp* ... *New Version*	95	84
Personal Consultant Easy ... *New Version*	495	435
Personal Consultant Image	495	435
Personal Consultant Online	995	869
Personal Consultant Plus ... *New Version*	2950	2589
Personal Consultant Runtime	95	84

ada language

AdaVantage *GSA-Validated by Meridian Software*	795	735
AdaVantage Utility Packages	50	47
DOS Environment Package	50	47
Janus/ADA C Pak *by R&R Software*	95	84
Janus/ADA D Pak *by R&R Software*	1250	1059
Janus/ADA ED Pak *by R&R Software*	395	349

apl language

APL*PLUS PC *by STSC*	695	495
APL*PLUS PC Spreadsheet Mgr *by STSC*	195	139
APL*PLUS PC Tools Vol 1 *by STSC*	295	199
APL*PLUS PC Tools Vol 2 *by STSC*	85	58
APL*PLUS PS/2 *by STSC* ... *New*	695	495
ATLAS*GRAPHICS *by STSC*	450	329
Financial/Statistical Library *by STSC*	275	189
Pocket APL *by STSC*	95	69
STATGRAPHICS *by STSC*	895	649

assembly language

386 ASM/LINK *Cross Asm by Phar Lap*	495	389
8088 Assembler *w/Z-80 Translator by 2500 AD*	100	89
ASMLIB *Function Library by BCSoft*	149	125
asmTREE *B-Tree Dev System by BCSoft*	395	329
Cross Assemblers *Various by 2500 AD*	CALL	CALL
EZASM *by C Source*	70	59
Microsoft Macro Assembler	150	93
Turbo Debugger *by Speedware*	89	79
Turbo Editasm *by Speedware*	99	84
Visible Computer: 8088 *by Software Masters*	80	64

basic language

db/Lib *for QuickBASIC by AJS Publishing*	139	119
Finally *by Komputerwerk*	99	85
MACH 2 *by Micro Help*	69	55
Microsoft QuickBASIC	99	63
QBase *Relational Database by Crescent*	89	79
Quick-Tools *by BCSoft*	130	109
QuickPak *by Crescent Software*	69	59
Scientific Subroutine Library *by Peerless*	125	99
Screen Sculptor *by Software Bottling*	125	91
Stay-Res *by MicroHelp*	69	55
True Basic *w/Run-time* ... *Special Price*	200	99
True Basic	100	79
Run-time Module	100	79
Various Utilities	50	41
Turbo BASIC Compiler *by Borland Intl*	100	64

blaise products

ASYNCH MANAGER *Specify C or Pascal*	175	135
C TOOLS PLUS/5.0 ... *New Version*	129	99
KeyPlayer *Super Batch Program* ... *New*	50	45
LIGHT TOOLS *for Datalight C*	100	65
PASCAL TOOLS	125	95
PASCAL TOOLS 2	100	79
PASCAL TOOLS & TOOLS 2	175	135
RUNOFF *Text Formatter*	50	45
TURBO ASYNCH PLUS	100	79
TURBO C TOOLS	129	99
TURBO POWER TOOLS PLUS	100	79
VIEW MANAGER *Specify C or Pascal*	275	199

borland products

EUREKA *Equation Solver*	167	105
Reflex: The Analyst	150	99
Sidekick	85	57
Superkey	100	64
Turbo Basic Compiler	100	64
Turbo Basic Database Toolbox	100	64
Turbo Basic Editor Toolbox	100	64
Turbo Basic Telecom Toolbox	100	64
Turbo C Compiler *(Call for support products)*	100	64
Turbo Lightning and Word Wizard	150	94
Turbo Lightning	100	64
Turbo Lightning Word Wizard	70	47
Turbo Pascal and Tutor ... *New Version*	CALL	CALL
Turbo Pascal ... *New Version*	100	64
Turbo Pascal Tutor ... *New Version*	70	41
Turbo Pascal Database Toolbox ... *New Version*	100	64
Turbo Pascal Editor Toolbox ... *New Version*	100	64
Turbo Pascal Gameworks Toolbox ... *New Version*	100	64
Turbo Pascal Graphix Toolbox ... *New Version*	100	64
Turbo Pascal Numerical Methods Toolbox ... *New*	100	64
Turbo Prolog Compiler	100	64
Turbo Prolog Toolbox	100	64

c compilers

C86PLUS *by Computer Innovations*	497	359
DeSmet C *w/Debugger & Large case*	209	184
DeSmet C *w/Debugger only*	159	138
Eco-C *Complete System by Ecosoft*	140	119
Instant C *by Rational Systems*	495	369
Instant-C/16M *by Rational Systems* ... *New*	895	695
Lattice C Compiler *vers. 3.2 from Lattice*	500	265
Mark Williams Let's C w/csd	75	54
Microsoft C Compiler w/CodeView ... *New Version*	450	269
Microsoft QuickC Compiler ... *New*	99	63
Optimum-C *by Datalight*	139	95
Turbo C Compiler *by Borland*	100	64
Uniware 68000/10/20 Cross Compiler *by SDS*	995	829

c interpreters

C-terp *by Gimpel, Specify compiler*	298	219
C Trainer *with Book by Catalytix*	122	87
Introducing C *by Computer Innovations*	125	99
Run/C *by Age of Reason*	120	69
Run/C Professional *by Age of Reason*	250	145

c utilities

Blackstar C Library *by Sterling Castle* ... *New*	125	98
C++ *by Guidelines w/version 1.1 kernel*	195	172
c-tree & r-tree Combo *by FairCom*	650	519
c-tree *ISAM File Manager*	395	315
r-tree *Report Generator*	295	239
Csharp Realtime Toolkit *by Systems Guild*	600	489
Curses Window Dev Pkg *by Aspen Scientific* ... *New*	119	105
with Source Code	289	249
dBx *dBASE to C Translator by Desktop AI*	350	299
with Source Code	550	419
Flash-up Windows *by Software Bottling*	90	78
GraphiC *Color version by Sci Endeavors*	350	274
GRAFLIB *by Sutrasoft*	175	159
HALO *Graphics by Media Cybernetics*	300	205
HALO Development Pkg *for Microsoft*	595	389
The HAMMER *by QES Systems*	195	129
PANEL *Forms Management by Roundhill*	295	215
PANEL/TC *for Turbo C by Roundhill*	129	95
PANEL Plus *by Roundhill*	495	395
PC Lint *by Gimpel Software*	139	99
PLOTHP *by Sutrasoft*	175	159
RTC PLUS *Fortran to C by Cobalt Blue*	450	399
Sapiens V8 *Virtual Memory Manager* ... *New*	300	265
Scientific Subroutine Library *by Peerless*	175	135
TE Text Editor *source by Sub Systems*	95	85
Vitamin C *by Creative Programming*	225	149
VC Screen *Forms Designer*	100	79
Zview *by Data Mgmt Consultants*	245	139

cobol language

COBOLspit *by Flexus*	395	329
FPLIB *for Realia COBOL by BCSoft*	149	129
Micro Focus COBOL *See Micro Focus Section*		
Microsoft COBOL *See Microsoft Section*		
PCDT *by Pro-Code*	995	895
Realia COBOL with RealMENU	1145	899
Realia COBOL	995	783
RealCICS	995	783
RM/COBOL *by Ryan-McFarland*	950	639
RM/COBOL 85 *by Ryan-McFarland*	1250	895
RM/WET+5 *by Ryan-McFarland* ... *New*	300	259
RM/Screens ... *New*	395	334
SCREENIO *by Norcom*	400	379
screenplay *for COBOL by Flexus*	175	129

css products

Combo Package *by Custom Software Systems*	199	175
PC/SPELL *Spelling Checker*	49	45
PC/TOOLS *UNIX-like Utilities*	49	45
PC/Vi *vi Editor*	149	99

debuggers & profilers

386 DEBUG *Cross Debugger by Phar Lap*	195	129
Advanced Trace-86 *by Morgan Computing*	175	115
Codesmith-86 *by Visual Age*	145	98
DSD87 *by Soft Advances*	125	79
MiniProbe *by Atron*	395	369
Periscope I *with Board by Periscope*	345	275
Periscope II *with NMI Breakout Switch*	175	139
Periscope II-X *Software only*	145	105
Periscope III *8 MHz version*	995	795
Periscope III *10 MHz version*	1095	875
The PROFILER *with Source Code by DWB*	125	89
TURBOsmith *Source debugger for Turbo Pascal*	99	89
The WATCHER *Profiler by Stony Brook*	80	51

disk utilities

Back-It *by Gazelle Systems* ... *New Version*	130	115
Disk Optimizer *by Softlogic Systems*	60	55
Disk Technician *by Prime Solutions* ... *New*	100	89
FASTBACK *by 5th Generation Systems*	179	129
Vcache *by Golden Bow Systems*	50	47
Vopt *by Golden Bow Systems*	50	47
Vfeature *by Golden Bow Systems*	80	74
Vfeature Deluxe *by Golden Bow Systems*	120	111
XenoCopy-PC *by XenoSoft*	80	69

dos utilities

Advanced Norton Utilities	150	99
Command Plus *by ESP Software*	80	69
Desqview *from Quarterdeck*	100	85
FANSI-CONSOLE *by Hersey Micro*	75	62
Mace Utilities *Paul Mace Software* ... *New*	99	89
MicroHelp Utility *by MicroHelp*	59	49
Norton Commander *by Peter Norton*	75	55
Norton Utilities *by Peter Norton*	100	59
OPAL *Shell Language by Software Factory*	99	89
Q-DOS II *by Gazelle Systems*	70	59
Taskview *by Sunny Hill Software*	80	55

essential products

C Utility Library	185	119
Essential Comm Library *with Debugger*	250	189
Essential Comm Library *Software Only*	185	125
Breakout Debugger Only *Any language*	125	89
Essential Graphics	250	183

forth language

CFORTH *Native Code Compiler by LMI*	300	229
FORTH/83 Metacompiler *Specify Target*	750	595
PC/FORTH *by Laboratory Microsystems*	150	109
PC/FORTH+ *by Laboratory Microsystems*	250	199
Programmer's Package #1 *by LMI* ... *New*	250	199
Programmer's Package #2 *by LMI* ... *New*	350	279
Programmer's Package #3 *by LMI* ... *New*	500	399
UR/FORTH *Also Available for OS/2 by LMI*	350	279
UR/FORTH Libraries	500	395

fortran language

50 MORE: FORTRAN *by Peerless Scientific*	125	95
ACS Time Series *by Alpha Computer Service*	495	389
AUTOMATED PROGRAMMER *by KGK Automated*	995	949
Essential Graphics *by Essential Software*	250	183
Forlib-Plus *by Alpha Computer Service*	70	44
FORTLIB *by Sutrasoft*	125	109
FORTRAN Addendum *by Impulse Engr*	95	85
FORTRAN Addenda *by Impulse Engr*	165	138
GRAFLIB *by Sutrasoft*	175	159
HALO *Graphics by Media Cybernetics*	300	205
I/O PRO w/No Limit Library *by MEF*	250	219
Microcompatibles Combo Package	240	215
Grafmatic	135	117
Plotmatic	135	117
Microsoft FORTRAN *w/CodeView*	450	269
No Limit Library *by MEF Environmental*	129	109
Numerical Analyst *by MAGUS*	295	249
PANEL *by Roundhill Computer Systems*	295	215
PLOTHP *by Sutrasoft*	175	159
RM/FORTRAN *by Ryan-McFarland* ... *New Version*	599	399
RTC PLUS *Fortran to C by Cobalt Blue*	450	399
Scientific Subroutine Lib *by Peerless*	175	135
Statistician *by Alpha Computer Service*	295	235
STATLIB.GL: *by Peerless*	295	239
STATLIB.TSF: *by Peerless*	295	239
Strings & Things *by Alpha Computer Service*	70	45

greenleaf products

Greenleaf C Sampler *for Turbo C & QuickC* ... *New*	95	69
Greenleaf Comm Library	185	125
Greenleaf Data Windows Library	225	155
with Source Code	395	249
Greenleaf Functions	185	125

help utilities

HELP/Control *by MDS*	125	99
On-line Help *from Dpt-Tech*	149	99
SoftScreen/HELP *by Dialectic Systems*	195	149

lattice products

Lattice C Compiler *ver 3.2 from Lattice*	500	265
with Library Source Code	900	495
C Cross Reference Generator	50	37
with Source Code	200	139
C-Food Smorgasbord *Function Library*	150	95
with Source Code	300	179
C-Sprite *Source Level Debugger* ... *Special Offer*	175	CALL
Curses *Screen Manager*	125	85
with Source Code	250	169
dBC III	250	169
with Source Code	500	356
dBC III Plus	750	594
with Source Code	1500	1184
LMK *Make Facility*	195	138
RPG II Combo *All three items below*	1100	875
RPG II Compiler *No Royalties*	750	625
SEU *Source Entry Utility*	250	199
Sort/Merge	250	199
Screen Design Aid Utility *for RPG II*	350	309
SecretDisk II *Encryption Utility* ... *New*	120	88
SideTalk *Resident Communications*	120	88
SSP/PC *Scientific Subroutine Library*	350	269
Text Management Utilities	120	88

metagraphics products

LightWINDOW/C *for Datalight C*	95	79
FontWINDOW	95	79
FontWINDOW/PLUS	275	229
MetaWINDOW *No Royalties*	195	159
MetaWINDOW/PLUS	275	229
TurboWINDOW/C *for Turbo C*	95	79
TurboWINDOW/Pascal *for Turbo Pascal*	95	79

micro focus products

Micro Focus COBOL/2	900	729
Micro Focus Level II COBOL *w/Animator*	495	395
Level II COBOL	349	279
Level II Animator	195	155
Micro Focus Level II COBOL/ET for UNIX	CALL	CALL
Micro Focus PC-CICS ... *New*	1495	1189
with Micro/SPF ... *New*	1595	1269
Micro Focus Personal COBOL	149	119
Micro Focus Professional COBOL	2000	1595
Micro Focus VS COBOL/XENIX	1495	1195
Micro Focus Support Products:		
COBOL/IQ *Ad hoc Report Writer*	495	395
COBOL/IQ *for DOS 3.X Networks*	995	795
FORMS-2	295	235
SOURCEWRITER	995	795

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

microport products

386 Unlimited License Kit	249	209
AT Unlimited License Kit	249	209
DOSMerge286 *Run DOS and UNIX together*	149	129
DOSMerge386 *Run DOS and UNIX together*	CALL	CALL
System V/386 Combination	799	699
386 Runtime System	199	169
386 Software Development System	499	429
Text Preparation System	199	169
System V/AT Combination	549	465
AT Runtime System	199	169
AT Software Development System	249	209
Text Preparation System	199	169

microsoft products

Microsoft BASIC Compiler *for XENIX*	695	419
Microsoft BASIC Interpreter *for XENIX*	350	209
Microsoft C Compiler *w/CodeView* ... *New Version*	450	269
Microsoft COBOL Compiler *with COBOL Tools* ... *for XENIX*	700	429
	995	609
Microsoft FORTRAN *Optimizing Compiler/CodeView*	450	269
Microsoft FORTRAN *for XENIX*	695	419
Microsoft Learning DOS	50	36
Microsoft MACH 10 *with Mouse & Windows*	549	369
Microsoft MACH 10 *Board only*	399	279
Microsoft Macro Assembler ... *New Version*	150	93
Microsoft Mouse *for IBM PS/2*	CALL	CALL
Microsoft Mouse *Bus Version*	CALL	CALL
Microsoft Mouse *Serial Version*	CALL	CALL
Microsoft Pascal Compiler	300	179
for XENIX	695	419
Microsoft QuickBASIC	99	63
Microsoft QuickC ... *New*	99	63
Microsoft Windows	99	63
Microsoft Windows Development Kit	500	299
Microsoft Word ... *New Version*	450	269

mks products

MKS AWK ... *New*	75	65
MKS RCS *Revision Control System* ... *New*	189	155
MKS Toolkit *with MKS Vi Editor*	139	109
MKS Trilogy *with AWK, CRYPT & Korn Shell* ... *New*	119	99
MKS Vi *Editor by MKS* ... *New*	75	65

modula-2 language

LOGITECH Modula-2 Development System ... *New*	249	199
Modula-2 Compiler Pack ... *New*	99	79
Modula-2 Toolkit ... *New*	169	139
LOGITECH Modula-2 ROM Pkg	299	239
LOGITECH Modula-2 Window Pkg	49	39
Macro2 *Macro preprocessor by PMI*	89	79
ModBase *by PMI*	89	79
ModGraph *by TEDNA* ... *New*	50	45
Repertoire *by PMI*	89	75
Science & Engrg Tools *by Quinn-Curtis*	75	67
Universal Graphics Library *by Quinn-Curtis*	130	119

mouse products

LOGIMOUSE BUS *with PLUS Pkg by LOGITECH*	119	98
with PLUS & PC Paintbrush	149	119
with PLUS & CAD Software	189	153
with PLUS & CAD & Paint	219	179
with First Publisher ... *New*	CALL	CALL
LOGIMOUSE C7 *with PLUS Pkg. Specify Connector*	119	98
with PLUS & PC Paintbrush	149	119
with PLUS & CAD Software	189	153
with PLUS & CAD & Paint	219	179
with First Publisher ... *New*	CALL	CALL
Microsoft Mouse *See Microsoft Section*		

other languages

ACTOR *by Whitewater Group* ... *New Version*	495	419
CCS MUMPS *Single-User by MGlobal*	60	50
CCS MUMPS *Single-User Multi-Tasking*	150	129
CCS MUMPS *Multi-User*	450	359
Marshal Pascal *by Marshal Language Systems*	189	165
Pascal-2 *by Oregon Software*	395	325
Personal REXX *by Mansfield Software*	125	99
SNOBOL4+ *by Catspaw*	95	80

other products

Carbon Copy Plus *by Meridian Technology*	195	159
Dan Bricklin's Demo Pgm *by Software Garden*	75	57
Dan Bricklin's Demo Tutorial	50	45
Fast Forward *by Mark Williams*	70	59
Instant Replay *by Nostradamus*	150	CALL
MicroTEX *Typesetting from Addison-Wesley*	295	CALL
Printer Drivers ... *New*	CALL	CALL
Net-Tools *by BCSoft*	149	129
Norton Guides *Specify Language* ... *New*	100	75
OPT-Tech Sort *by Opt-Tech Data Proc*	149	99
PC/TOOLS *by Custom Software*	49	45
Screen Machine *by MicroHelp*	79	59
SuperSort *by LifeStyle* ... *New*	139	119

phoenix products

Pasm86 *Macro Assembler version 2.0*	195	108
Pdisk *Hard Disk & Backup Utility*	145	99
Pfantasy Pac *Phoenix Combo*	995	595
Pfinish *Execution Profiler*	395	209
Pfix86plus *Symbolic Debugger*	395	209
PforCe *Specify C Compiler*	395	209
PforCe++ *Specify C Compiler and C++*	395	209
Plink86plus *Overlay Linker*	495	275
Pmaker *Make Utility*	125	78
Pmate *Macro Text Editor*	195	108
Pre-C *Lint Utility*	295	154
Ptel *Binary File Transfer Program*	195	108

polytron products

PolyBoost *Software Accelerator* ... *Special Price*	80	54
PolyDesk III	99	72
PolyDesk III Archivist	50	42
PolyDesk III Cryptographer	50	42
PolyDesk III Talk	70	52
PolyLibrarian *Library Manager*	99	89
PolyLibrarian II *Library Manager*	149	129
PolyMake *UNIX-like Make Facility*	149	129
PolyShell ... *Special Price*	149	105
Polytron C Beautifier	50	45
PolyXREF *Complete Cross Ref Utility*	219	185
PolyXREF *One language only*	129	109
PVCS Corporate *Version Control System*	395	329
PVCS Personal	149	129

program mgmt utilities

Interactive EASYFLOW *by Haventree* .. *New Version*	150	125
PrintQ *by Software Directions*	89	64
Quilt Computing Combo Package	250	199
QMake *Program Rebuild Utility*	99	79
SRMS *Software Revision Mgmt System*	185	159
Sapiens MAKE & VS ... *New*	439	379
Sapiens MAKE ... *New*	179	155
Source Print *by Aldebaran Labs*	97	75
TLIB *Version Control System by Burton*	100	89
Tree Diagrammer *by Aldebaran Labs*	77	67

raima products

dbQUERY *Single-User Query Utility*	195	129
Single-User with Source Code	495	389
Multi-User	495	389
Multi-User with Source Code	990	699
dbVISTA *Single-User DBMS*	195	129
Single-User with Source Code	495	389
Multi-User	495	389
Multi-User with Source Code	990	699

sco products

Complete XENIX System V *by SCO*	1295	994
Development System	595	499
Operating System *Specify XT or AT*	595	499
Text Processing Package	195	144
Lyrix *by SCO*	595	449
SCO Professional *1-2-3 Workalike for XENIX*	795	595
SCO XENIX-NET	595	495
XENIX System V 386 *by SCO* ... *New*	CALL	CALL

softcraft products

Btrieve *ISAM Mgr with No Royalties*	245	184
Xtrieve *Query Utility*	245	184
Report Option *for Xtrieve*	145	99
Btrieve/N *for Networks*	595	454
Xtrieve/N	595	454
Report Option/N *for Xtrieve/N*	345	269

text editors

Brief & dBrief Combo *from Solution Systems*	275	CALL
Brief	195	CALL
dBrief *Customizes Brief for dBASE III*	95	CALL
de *by David Livshin* ... *New*	75	65
Epsilon *Emacs-like editor by Lugaru*	195	147
KEDIT *by Mansfield Software*	125	98
Micro/SPF *by PHASER SYSTEMS*	175	139
Microsoft Word ... *New Version*	450	269
PC/VI *by Custom Software Systems*	149	99
SPF/PC *by Command Technology Corp*	CALL	CALL
Vedit Plus *by CompuView*	185	129

turbo pascal utilities

ALICE *Interpreter by Software Channels*	95	86
DOS/BIOS & Mouse Tools *by Quinn-Curtis*	75	67
Flash-up Windows *by Software Bottling*	90	78
MACH 2 *for Turbo Pascal by Micro Help*	69	55
MetraByte D/A Tools *by Quinn-Curtis*	100	89
Science & Engrg Tools *by Quinn-Curtis*	75	67
Screen Sculptor *by Software Bottling*	125	91
Speed Screen *by Software Bottling*	35	32
System Builder *by Royal American* ... *New Version*	150	129
IMPEX *Query Utility*	100	89
Report Builder ... *New Version*	130	115
TDebugPLUS *by TurboPower Software*	60	49
Tmark *by Tangent Designs*	80	69
Turbo EXTENDER *by TurboPower Software*	85	64
Turbo OPTIMIZER *by TurboPower*	75	65
with Source Code	125	98
Turbo Professional *by Sunny Hill*	70	45
TurboHALO *from IMSI*	129	98
TurboPower Utilities *by TurboPower*	95	78
TurboRef *by Gracon Services*	50	36
TURBOsmith *Source Debugger by Visual Age*	99	89
Universal Graphics Library *by Quinn-Curtis*	130	119

wendin products

Operating System Toolbox	99	79
PCMX *Operating system*	99	79
PCVMS *Similar to VAX/VMS*	99	79
Wendin-DOS *Multitasking DOS*	99	85
Wendin-DOS Application Developer's Kit ... *New*	99	85
XTC *Text Editor w/Pascal source*	99	75

xenix/unix products

Btrieve *ISAM File Mgr by SoftCraft*	595	454
C-terp *by Gimpel, Specify compiler*	498	379
c-tree *ISAM Mgr by FairCom*	395	315
dBx *with Library Source by Desktop AI*	550	419
DIRECTORY SHELL 286		
by American Mgmt Sys ... *New*	349	295
DIRECTORY SHELL 386		
by American Mgmt Sys ... *New*	495	415
DOSIX *Console Version by Data Basics*	400	349
DOSIX *User Version by Data Basics*	200	179
Micro Focus Products *See Micro Focus Section*		
Microport Products *See Microport Section*		
Microsoft Products *See Microsoft Section*		
PANEL Plus *by Roundhill Computer Systems*	795	535
REAL-TOOLS *Binary Version by PCT*	99	89
Library Source Version	599	539
Complete Source Version	999	729
RM/COBOL *by Ryan-McFarland*	1250	949
RM/FORTRAN *by Ryan-McFarland*	750	549
SCO Products *See SCO Section*		

Terms are subject to change.

©1987 Programmers Connection, Inc.

LOWEST PRICES

Due to printing lead times, some of our current prices may differ from those shown here. Call for latest pricing.

FREE SHIPPING

Orders within the USA (including Alaska & Hawaii) are shipped FREE via UPS. Express shipping is available at the shipping carrier's standard rate with no rush fees or handling charges. To avoid delays when ordering by mail, please call first to determine the exact cost of express shipping.

VISA and MasterCard are accepted at no extra cost. Your card is charged when your order is shipped. Mail orders please include credit card expiration date and authorized signature.

CODs AND POs

CODs and Purchase Orders are accepted at no extra cost. No personal checks are accepted on COD orders. POs with net 30-day terms (with initial minimum order of $100) are available to qualified US accounts only.

SALES TAX

Orders outside of Ohio are not charged state sales tax. Ohio customers please add 6% Ohio tax or provide proof of tax-exemption.

INTERNATIONAL ORDERS

Shipping charges for International and Canadian orders are based on the shipping carrier's standard rate. Since rates vary between carriers, please call or write for the exact cost. International orders (except Canada), please include an additional $10 for export preparation. All payments must be made with US funds drawn on a US bank. Please include your telephone number when ordering by mail. Due to government regulations, we cannot ship to all countries.

VOLUME ORDERS

Volume orders may qualify for additional discounts. Call us for special pricing.

SOUND ADVICE

Our knowledgeable technical staff can answer technical questions, assist in comparing products and send you detailed product information tailored to your needs.

30-DAY GUARANTEE

Most of our products (excluding books) come with a 30-day documentation evaluation period or a 30-day return guarantee. Please note that some manufacturers restrict us from offering guarantees on their products. Call for more information.

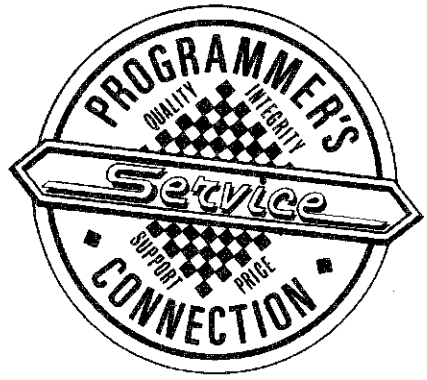
MAIL ORDERS

Please include your telephone number on all mail orders. Be sure to specify computer, operating system and any applicable compiler or hardware interface(s). Send mail orders to:

Programmer's Connection
7249 Whipple Ave. NW
North Canton, OH 44720

USA	800-336-1166
CANADA	800-225-1166
OHIO & ALASKA (Collect)	216-494-3781
TELEX	9102406879
EASYLINK	62806530
INTERNATIONAL	216-494-3781
CUSTOMER SERVICE	216-494-8899

Hours: Weekdays 8:30 AM to 8:00 PM EST.



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
                mov ax,cs               ;AX = CS
                mov ds,ax               ;DS = AX = CS
                mov es,ax               ;ES = AX = CS
                lea di,temp_buff        ;DI points to buffer
                mov cx,5                ;CX = 5 (for 10 digits)
                mov al,0                ;AL = 0
clear:          stosb                   ;store a 0
                loop clear              ;fill buffer with zeros
                mov cx,32               ;
                test dx,8000h           ;is low word negative?
                jz no_neg               ;no, skip code
                not dx                  ;DX = -DX
                not bx                  ;BX = -BX
                add bx,1                ;BX = BX + 1
                adc dx,0                ;adjust for carry
no_neg:         std                     ;set direction to back
l_loop:         push cx                 ;save counter
                mov cx,5                ;five bytes=10 digits
                lea si,temp_buff        ;SI points to buffer
                add si,4                ;SI points to buffer end
                mov di,si               ;DI = SI
                shl bx,1                ;grab a bit
                rcl dx,1                ;and then rotate
adjust:         lodsb                   ;load AL with byte
                adc al,al               ;add carry
                daa                     ;decimal adjust
                stosb                   ;store in temp_buff
                loop adjust             ;keep going
                pop cx                  ;restore counter
                loop l_loop             ;and do again
                cld                     ;set flag to forward
                lea di,lstr             ;DI points to lstr
                xor dx,dx               ;DX = 0
                lea si,temp_buff        ;SI points to buffer
                mov cx,10               ;CX = 10 characters
                xor bx,bx               ;BX = 0
x_str:          test cx,1               ;high or low?
                jnz y_str               ;low
                lodsb                   ;AL = char
                mov ah,al               ;AH = AL
                push cx                 ;save CX
                mov cl,4                ;CL = 4
                shr al,cl               ;shift AL right 4
                pop cx                  ;restore CX
                jmp z_str               ;do a character
y_str:          mov al,ah               ;low part
                and al,0Fh              ;and it
z_str:          test al,0Fh             ;is it zero?
                jnz w_str               ;no
                or bx,bx                ;a leading zero
                jz l_next               ;no
w_str:          inc bx                  ;BX has length
                or al,'0'               ;AL = ASCII char
                stosb                   ;store in string
l_next:         loop x_str              ;continue
                add bx,dx               ;add 0
                jnz l_done              ;are we done?
                inc bx                  ;bump length
                mov al,'0'              ;make 0
                stosb                   ;store it
l_done:         cld                     ;back to forward
                mov byte ptr [di],0     ;end of string byte
                pop ax                  ;restore AX
                mov es,ax               ;ES = DS
                lea si,lstr             ;SI points to lstr
                mov di,[bp+4]           ;DI points to str
                mov cx,12               ;we'll move all
```



```
                rep movsb               ;move them
                pop di                  ;restore
                pop si                  ; regs
                pop bx                  ;  from
                pop es                  ;   the
                pop ds                  ;    stack
                pop bp
                ret 4
longstr         endp


;;;;;;;;;;;;;;;
;hides the cursor
hide_cursor     proc near
                mov ah,1                ;cursor size function
                mov ch,20h              ;set it to 20h
                int 10h                 ;BIOS does it
                ret
hide_cursor     endp

;;;;;;;;;;;;;;;
;saves the DOS cursor for later restoration
save_cursor     proc near
                jmp get_it              ;jump past data
save_curs dw 0
get_it:         mov ah,3                ;get cursor size function
                mov bh,0                ;page is 0
                int 10h                 ;BIOS does it
                mov cs:save_curs,cx     ;CX has size - save it
                ret
save_cursor     endp

;;;;;;;;;;;;;;;
;restores the saved DOS cursor
rest_cursor     proc near
                mov cx,cs:save_curs     ;retrieve cursor size
                mov ah,1                ;set cursor size function
                int 10h                 ;BIOS does it
                ret
rest_cursor     endp

toolcode    ends
            end
```

```
{$Z63,S3,V+,E1,W-,F1,T0}
definition module sheltool(output);
(* copyright 1987, John J. Newlin *)
const
  main_color = 16#0E#;
  curs_color = 16#70#;
  maxfiles = 200;
  up     = 72;   f1 = 59;
  down   = 80;   f2 = 60;
  endkey = 79;   f3 = 61;
  home   = 71;   f4 = 62;
  retkey = 28;   f5 = 63;
  pgup   = 73;   f6 = 64;
  pgdn   = 81;   f7 = 65;
  spce   = 57;   f8 = 66;
  esc    = 1;    f9 = 67;
  tab    = 15;  f10 = 68;
  ins    = 82;
  del    = 83;
  ctrl_end =  117;
  ctrl_home = 119;
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

LEARN about . . .

Neural Nets

A major upheaval in thinking about artificial intelligence is underway. Interest is accelerating in the "neural net" approach, which is loosely based on human brain research. A tutorial introduction to this exciting new technology is now available in a model called Netwurkz (tm).

Netwurkz eliminates the complicated math found in "unsupervised learning" so that underlying principles may be grasped quickly. The network "learns" when you add new DATA statements (interconnects), or modify existing ones. A trace on/off feature prints the neural net computation in progress. List based internal representation yields speed.

Netwurkz is intended for people not familiar with neural modeling who want to maintain or broaden their technical skills and background. Source for all neural net software and an object-only copy of the PL/D compiler are included

Compilers

PL/D (tm) is a new system language that contains documentation so definitive that your understanding of all languages is enhanced. Like other languages, PL/D is described in narrative form, alphabetically by token and via flow diagrams. However, PL/D also comes with self-compiling source! If you have a question after reading the manual, you can look at the source. If you do not accept the answer, you can change it by modifying the code.

What you learn from PL/D also applies to all compilers: symbol table organization, macro handling, nested structure implementation, cross reference listing, code generation, etc. You will study an unusually efficient implementation: PL/D is 2-3 times faster than the recent "fast" C compilers.

If you are a bright self-sufficient type who wants to master your development tools, PL/D is for you.

PC or compatible with 256K and DOS 2.0 or above and 5.25 inch diskette required.

DAIR Computer Systems 3440 Kenneth Drive, Palo Alto CA 94303
(415) 494-7081

Name: _____
Company: _____
Address: _____
City, State, Zip: _____
Phone: _____
VISA, M/C: _____ Expire: _____

Netwurkz: neural net demo	$79.95	
PL/D: fast self-compiles	$124.95	
Both of above	$154.95	
Sales Tax, CA only		
Ship overseas	$10.00	
TOTAL		

Want to know more?
(415) 494-7081

Available Now

Number 5 on the Reader Service Card

386•386•386•386•386•386•386•386•386•386•386•386•386•386•386•386•386•386•386•386•386•

NOW!

386 C and Pascal for MS-DOS

MetaWare Incorporated announces the *first* available C and Pascal compilers that generate

protected-mode 80386 code

for running on any 80386 machine that runs MS-DOS (e.g., the Compaq Deskpro 386). The compilers are functionally identical to the well-respected 8086/286 MS-DOS High C™ and Professional Pascal™ compilers that have received outstanding reviews in such magazines as Computer Language, Dr. Dobb's, and PC Tech Journal. Our compilers are currently used by industry leaders such as Ashton-Tate, AutoDesk, ANSA, and Lifetree. Now you can get them generating 80386 code.

If you have an application that requires the large 32-bit address space and the full 32-bit registers of the 80386, expand your marketplace to the rapidly growing supply of 80386 MS-DOS machines. Contact MetaWare for your 80386 software solution today!
(408) 429-6382, telex 493-0879.

Durable Software Constructed Automatically™

MetaWare™ INCORPORATED

903 Pacific Avenue, Suite 201 • Santa Cruz, CA 95060-4429

•386•386•386•386•386•386•386•386•386•386•386•386•386•386•386•386•386•386•386•386•

Number 45 on the Reader Service Card

Introducing

Quaid Analyzer

the tool that created CopyWrite

Now you can debug your own programs with a professional quality debugger - the one that unraveled every form of copy-protection used on the PC.

With the Quaid Analyzer, you can:

- ☐ See occurences of any interupt, with its meaning shown on the screen.
- ☐ View memory as text or instructions, scrolling as easily as you do with an editor.
- ☐ Run until a memory location or I/O port is changed.
- ☐ Protect your hard disk from accidental destruction.
- ☐ Analyze software without the source, even when it uses countermeasures to thwart tracing.
- ☐ See all stages of the boot load.

We kept the Quaid Analyzer off the market to avoid helping publishers with copy-protection. Now that copy-protection is gone, we can sell it to you.

The Quaid Analyzer is a software tool occupying 100K bytes. It runs on any IBM PC and most MS-DOS systems without hardware modification.

To order Call **(416) 961-8243**

Quaid Analyzer $99 U.S.

All orders shipped at our expense within a day. All major credit cards accepted.

or return coupon to:
45 Charles St. East
Third Floor, Dept. 603
Toronto, Ontario. M4Y 1S2

Payment method MC-Visa-Amex-Diners-Check
Card No. _____
Expiry Date _____
Name _____
Address _____
City/State _____
Phone No. _____
Signature _____

Quaid Software Limited

Ask about Disk Explorer the program that takes over where Quaid Analyzer leaves off.

Contact this Advertiser directly

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
  ctrl_pgup = 132;
  horiz     = CHR(196);
  vert      = CHR(179);
  ul        = CHR(218);
  ur        = CHR(191);
  ll        = CHR(192);
  lr        = CHR(217);
  space     = CHR(32);
  windows   = 2;

type
  str8 = string[8];
  str12 = string[12];
  str64 = string[64];
  regtype = record
              ax,bx,cx,dx,di,si,ds,es,flags : integer;
            end;
  {$U0}
  buff_type = record
                dos_use : array[0..20] of char;
                attr : char;
                time,date,losize,hisize : integer;
                filename  : str12;
              end;

  file_type = record
                attr : char;
                time,date,losize,hisize : integer;
                name : str12;
                desig : integer;
              end;
  file_array = array[1..maxfiles] of file_type;
  dir_array = array[1..windows] of file_array;
  {$U1}

  longint = array[0..1] of integer;
  longstring = string[11];
  screenbuff_type = array[1..4000] of char;

var
{  files1,files2 : file_array; }
  files : dir_array;
  save_t_time,save_time,attr,seed : integer;
  entry_str : string;
  color : boolean;
  screenbuff : screenbuff_type;
function shiftl(target,bits : integer) : integer;
function upcase(ch : char) : char;
function shiftr(target,bits : integer) : integer;
function hi(target : integer) : integer;
function lo(target : integer) : integer;
procedure exec(var name : string);
procedure execute(var command : string);
function format_num(long : longint; width : integer) : string;
procedure get_files(var mask : string; var files : file_array;
                    var count : integer);
function filedate(code : integer) : str12;
procedure draw_box(col,row,width,depth : integer);
procedure save_cursor;
procedure hide_cursor;
procedure rest_cursor;
function filetime(code : integer) : str12;
function convert(var st : str12) : str12;
procedure sort_files(var files : file_array; var items : integer);
procedure longstr(var long : longint; var strng : longstring);
procedure addlong(var total,n1,n2 : longint);
function delete_file(addr : integer) : integer;
```



```
function keycode(var status,ascii,scan : integer) : boolean;
procedure scroll(ulx,uly,lrx,lry,lines,attr,dir : integer);
procedure savebox(col,row,width,depth,attr : integer);
procedure restbox(col,row,width,depth,attr : integer);
procedure set_dta(var buffer : buff_type);
procedure init_screen;
procedure msdos(var regs : regtype);
procedure setxy(col,row : integer)
procedure screenwrite(col,row,attr : integer; var str : string);
procedure fillstr(var str : string; num : integer; ch : char);
procedure move(v1addr,v2addr,bytes : integer);
function chdir(var dirname : string) : integer;
function mkdir(var dirname : string) : integer;
function rmdir(var dirname : string) : integer;
function findfirst(var pathname : string; attr : integer) : integer;
function findnext : integer;
procedure cls(attribute : integer);
procedure strng(num : integer; var numstr : string);
function abs_read(drive,sectors,start,buff_addr:integer):integer;
procedure getdir(var path : string);
function set_mem : integer;
function video_mode : integer;
procedure fx(barlen,battr,col,row,attr : integer; var str : string);
end.

{$Z63,S3,V+,E1,W-,F1,T0}
(* Copyright 1987, John J. Newlin *)
implementation module sheltool(input,output);
function shiftl(target,bits : integer) : integer; external;
function shiftr(target,bits : integer) : integer; external;
function hi(target : integer) : integer; external;
function lo(target : integer) : integer; external;
function upcase(ch : char) : char; external;
procedure exec(var name : string); external;
function delete_file(addr : integer) : integer; external;
procedure longstr(var long : longint; var strng : longstring); external;
procedure save_cursor; external;
procedure hide_cursor; external;
procedure rest_cursor; external;
procedure addlong(var total,n1,n2 : longint); external;
function keycode(var status,ascii,scan : integer) : boolean; external;
procedure scroll(ulx,uly,lrx,lry,lines,attr,dir : integer); external;
procedure savebox(col,row,width,depth,attr : integer); external;
procedure restbox(col,row,width,depth,attr : integer); external;
procedure set_dta(var buffer : buff_type); external;
procedure init_screen; external;
procedure msdos(var regs : regtype); external;
procedure setxy(col,row : integer);  external;
procedure screenwrite(col,row,attr : integer; var str : string); external;
procedure fillstr(var str : string; num : integer; ch : char); external;
procedure move(v1addr,v2addr,bytes : integer); external;
function chdir(var dirname : string) : integer; external;
function mkdir(var dirname : string) : integer; external;
function rmdir(var dirname : string) : integer; external;
procedure getdir(var path : string); external;
function findfirst(var pathname : string; attr : integer) : integer; external;
function findnext : integer; external;
procedure cls(attribute : integer); external;
procedure strng(num : integer; var numstr : string); external;
function abs_read(drive,sectors,start,buff_addr:integer):integer; external;
function set_mem : integer; external;
function video_mode : integer; external;
procedure execute(var command : string);
var l : integer;
begin
  l := length(command);
  command := concat(" ",command," ");
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Software Tools

❑ **The DATE UTILITIES** let you copy, erase, or display files based on their name **and** date. You can also change a file's time stamp. All utilities support wildcards for file names and let you limit the files chosen to those created **before, on,** or **after** a specified date. The package includes DCOPY, DERASE, DDIR, CDATE, and three other general purpose utilities. Version 1.5 costs $47.50.

❑ **PROINDEX** is a professional document indexing package. It works with all word processed text to create an index in any style. PROINDEX compares your text file to a **list** of keywords or phrases. These keywords **or** phrases can be up to 70 characters long! No limit on the size of the file to index or the number of references. Version 2.24 costs $89.95.

❑ **Soft Fonts** for the HP LaserJet and compatible printers. We offer **inexpensive** Helv, Roman, and Script font families. We have Roman fixed width, Bar code 39, Title fonts, Symbol sets, and Credit cards. All font families are $50 or less! Printer drivers for most word processors cost only $15 per font family! Custom fonts & a download utility are also available.

All tools work on the IBM PC or any compatible, MS-DOS 2.1 & up. No charge for shipping in the U.S. Thirty day money back guaranty!

ECI P.O. Box 61, Wasco, Illinois 60183

VISA 312-377-3520 MC

Number 70 on the Reader Service Card

Make your C language programs memory resident with DMS RESIDENT-C

Lattice 3.0 Microsoft 4.0

"hot-key" enable

$79.95 $149.95 w/source

Make your assembler programs memory resident with DMS RESIDENT-ASM

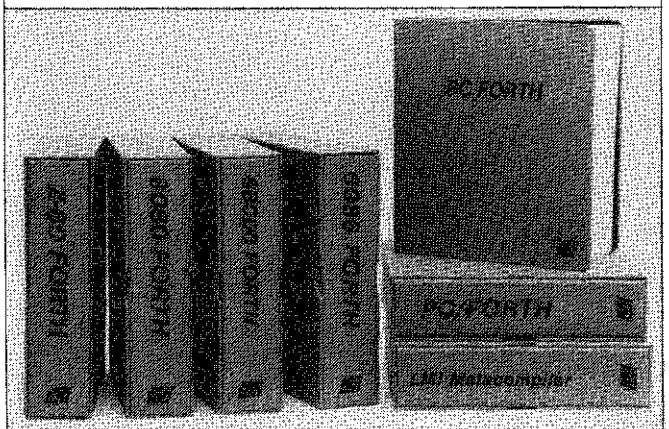
"hot-key" enable

$79.95 $149.95 w/source

American Software International
P.O. Box 523
Windsor, Ct 06095-9998
(203)688-5054

Number 151 on the Reader Service Card

TOTAL CONTROL with *LMI FORTH*™



For Programming Professionals:

an expanding family of compatible, high-performance, Forth-83 Standard compilers for microcomputers

For Development: Interactive Forth-83 Interpreter/Compilers

- 16-bit and 32-bit implementations
- Full screen editor and assembler
- Uses standard operating system files
- 400 page manual written in plain English
- Options include software floating point, arithmetic coprocessor support, symbolic debugger, native code compilers, and graphics support

For Applications: Forth-83 Metacompiler

- Unique table-driven multi-pass Forth compiler
- Compiles compact ROMable or disk-based applications
- Excellent error handling
- Produces headerless code, compiles from intermediate states, and performs conditional compilation
- Cross-compiles to 8080, Z-80, 8086, 68000, 6502, 8051, 8096, 1802, and 6303
- No license fee or royalty for compiled applications

For Speed: CForth Application Compiler

- Translates "high-level" Forth into in-line, optimized machine code
- Can generate ROMable code

Support Services for registered users:

- Technical Assistance Hotline
- Periodic newsletters and low-cost updates
- Bulletin Board System

Call or write for detailed product information and prices. Consulting and Educational Services available by special arrangement.

Laboratory Microsystems Incorporated
Post Office Box 10430, Marina del Rey, CA 90295
Phone credit card orders to: ***(213) 306-7412***

Overseas Distributors.
Germany: Forth-Systeme Angelika Flesch, Titisee-Neustadt, 7651-1665
UK: System Science Ltd., London, 01-248 0962
France: Micro-Sigma S.A.R.L., Paris, (1) 42.65.95.16
Japan: Southern Pacific Ltd., Yokohama, 045-314-9514
Australia: Wave-onic Associates, Wilson, W.A., (09) 451-2946

Number 63 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
  command[1] := chr(1);
  command[length(command)] := chr(13);
  if length(command) > 126 then return;
  savebox(1,1,80,25,address(screenbuff));
  cls(15);
  rest_cursor;
  setxy(1,1);
  exec(command);
  hide_cursor;
  restbox(1,1,80,25,address(screenbuff));
end;

procedure draw_box(col,row,width,depth : integer);
var x,y : integer;
    side : string;
begin
  fillstr(side,width-2,horiz[1]);
  side := concat(ul,side,ur);
  screenwrite(col,row,main_color,side);
  fillstr(side,width-2,space[1]);
  side := concat(vert,side,vert);
  for y := row+1 to row+depth-1 do screenwrite(col,y,main_color,side);
  fillstr(side,width-2,horiz[1]);
  side := concat(ll,side,lr);
  screenwrite(col,row+depth,main_color,side);
end;

procedure fx(barlen,battr,col,row,attr : integer; var str : string);
begin
  if barlen < length(str) then
    begin
      screenwrite(col,row,attr,str);
      return;
    end
  else
    begin
      while length(str) < barlen do str := concat(str," ");
      screenwrite(col,row,battr,str);
    end;
end;

procedure get_files(var mask : string; var files : file_array;
                    var count : integer);
var dir : buff_type;
begin
  set_dta(dir);
  count := 0;
  if findfirst(mask,16#1F#) = 0 then          {attr bit pattern = 00010111}
     begin
       if dir.filename[1] <> '.' then
         begin
           count := succ(count);
           move(address(dir.attr),address(files[count]),22);
           files[count].desig := 0;
         end;
     end;
   while (count < maxfiles) and (findnext = 0) do
     begin
       if dir.filename[1] <> '.' then
         begin
           count := succ(count);
           move(address(dir.attr),address(files[count]),22);
           files[count].desig := 0;
         end;
     end;
end;
```



```
function filedate(code : integer) : str12;
var i,y,m,d : integer;
    ys,ms,ds : str12;
begin
  y := hi(code);
  y := shiftr(y,1) + 80;
  if y > 99 then y := y - 100;
  strng(y,ys);
  m := shiftr(code,1);
  m := lo(m);
  m := shiftr(m,4);
  strng(m,ms);
  if length(ms) = 1 then ms := concat("0",ms);
  d := shiftl(code,3);
  d := lo(d);
  d := shiftr(d,3);
  strng(d,ds);
  if length(ds) = 1 then ds := concat("0",ds);
  filedate := concat(ms,"/",ds,"/",ys);
end;

function filetime(code : integer) : str12;
var h,m : integer;
    hr,mi,x : str12;
begin
  h := hi(code);
  h := shiftr(h,3);
  if h >= 12 then
    begin
      if h > 12 then h := h - 12;
      x := ' p.m.';
    end else x := ' a.m.';
  strng(h,hr);
  if length(hr) = 1 then hr := concat("0",hr);
  m := shiftr(code,6);
  m := lo(m);
  m := shiftl(m,3);
  m := lo(m);
  m := shiftr(m,2);
  strng(m,mi);
  if length(mi) = 1 then mi := concat("0",mi);
  filetime := concat(hr,":",mi,x);
end;

function convert(var st : str12) : str12;
var n,i : integer;
    name : string[13];
begin
  n := pos(".",st);
  if (n > 0) and (n <> 9) then
    begin
      name := '            ';
      move(address(st[1]),address(name[1]),n-1);
      move(address(st[n]),address(name[9]),length(st)-n+1);
    end
  else name := st;
  name[9] := chr(32);
  while length(name) < 12 do name := concat(name," ");
  convert := name;
end;

procedure sort_files(var files : file_array; var items : integer);
var jump,i,j : integer;
    done : boolean;
    temp : file_type;
begin
  jump := items;
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
  while jump > 1 do
    begin
      jump := jump div 2;
      repeat
        done := true;
        for j := 1 to items - jump do
          begin
            i := j + jump;
            if files[j].name > files[i].name then
              begin
                temp := files[j];
                files[j] := files[i];
                files[i] := temp;
                done := false;
              end;
          end;
      until done;
    end;
end;

function format_num(long : longint; width : integer) : string;
var str : longstring;
     n,i,temp : integer;
begin
  longstr(long,str);
  n := length(str);
  if n in [4..6] then insert(",",str,n-2);
  if n in [7..9] then
    begin
      insert(",",str,n-5);
      insert(",",str,n-1);
    end;
  n := length(str);
  if width > n then for i := 1 to (width - n) do str := concat(" ",str);
  format_num := str;
end;

begin
  entry_str := '';
  color := video_mode <> 7;
  if color then attr := 16#0B# else attr := 16#0F#;
  init_screen;
end.

{$Z63,S3,V+,E1,W-,F1,T0}
(*  copyright 1987, John J. Newlin
    Z63 = full optimization
     S3 = allow Pascal extensions
     V+ = allow variable length strings
     E1 = use actual procedure names for linking
     W- = suppress warnings about unused variables
     F1 = optimize for speed
     T0 = do not generate symbol table info
*)

program shell(input,output);
import sheltool;

const
  win1_beg = 5;
  win2_beg = win1_beg + 10;
  win_col = 2;
  win3_beg = 5;
  win3_col = 61;

var
  paragraphs,action,code,i : integer;
```



```
  total,count,x,y,curr_page,last_page,index : array[1..windows] of integer;
  root_dir,current_dir,default_dir,str : string;
  dir : array[1..windows] of str64;
  beg_y,max_y : array[1..windows] of integer;
  copy_flag,window_flag : boolean;
  key,last_drive : char;
  drive_list : array[1..26] of char;

procedure terminate;
begin
  code := chdir(default_dir);
  rest_cursor;
  cls(15);
  halt;
end;

procedure rename_file(oldfile,newfile : string);
var f : text;
begin
  reset(f,oldfile);
  close(f);
  rename(f,newfile);
end;

function user_entry(prompt : string) : string;
var temp,blank : string;
    i : integer;
begin
  screenwrite(4,2,main_color,prompt);
  setxy(4,3);
  rest_cursor;
  readln(temp);
  hide_cursor;
  fillstr(blank,70,chr(32));
  screenwrite(4,2,main_color,blank);
  screenwrite(4,3,main_color,blank);
  for i := 1 to length(temp) do temp[i] := upcase(temp[i]);
  user_entry := temp;
end;

procedure get_drive_list;
var regs : regtype;
    i : integer;
begin
  regs.ax := 16#0E00#;
  regs.dx := ord(current_dir[1]) - 65;
  msdos(regs);
  last_drive := chr(lo(regs.ax) + 64);
  for i := 65 to ord(last_drive) do drive_list[i-64] := chr(i);
  drive_list[ord(last_drive)-63] := chr(0);
end;

procedure copy_file(index,win : integer; var files : file_array);
var cmd : string;
    dest : integer;
begin
  if win = 1 then dest := 2 else dest := 1;
  cmd := concat('COPY ',files[index].name," ",dir[dest],' > NUL');
  cmd := concat(" ",cmd," ");
  cmd[length(cmd)] := chr(13);
  cmd[1] := chr(length(cmd));
  exec(cmd);
  copy_flag := true;
end;

procedure scroll_it(y,lines,dir : integer);
begin
```


PRODUCED 2024 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
  scroll(3,y,46,y+7,lines,main_color,dir);
end;

procedure drive_menu;
var i,code,index,last,keystat,ascii,scan : integer;
        str : string;

function drive_str(indx : integer) : string;
begin
  drive_str := 'Drive  ';
  drive_str[7] := drive_list[indx];
end;

begin
  last := ord(last_drive) - 64;
  draw_box(win3_col,win3_beg,10,last+1);
  scroll(win3_col+1,win3_beg+1,win3_col+8,win3_beg+last,last,main_color,0);
  for i := 1 to last do
    begin
      str := drive_str(i);
      screenwrite(win3_col+1,i+win3_beg,main_color,str);
    end;
  index := ord(current_dir[1]) - 64;
  loop
    str := drive_str(index);
    fx(8,curs_color,win3_col+1,index+win3_beg,main_color,str);
    repeat until keycode(keystat,ascii,scan);
    if scan = 1 then terminate;
    if scan = 28 then
      begin
        str[1] := drive_list[index];
        str[2] := ':';
        str[3] := chr(0);
        code := chdir(str);
        return;
      end;
    if (scan = down) then
      begin
        fx(0,curs_color,win3_col+1,index+win3_beg,main_color,str);
        if index < last then index := succ(index)
          else if index = last then index := 1;
      end;
    if (scan = up) then
      begin
        fx(0,curs_color,win3_col+1,index+win3_beg,main_color,str);
        if index > 1 then index := pred(index)
          else if index = 1 then index := last;      end;
    if scan = tab then return;
  end;
end;

procedure top_line(y : integer; var dir : str64);
var line : string;
    i : integer;
begin
  fillstr(line,44,chr(196));
  for i := 1 to length(dir) do line[i+2] := dir[i];
  screenwrite(win_col+1,y,main_color,line);
end;

function show(index:integer; var files : file_array) : string;
var ftime,fdate : string[14];
    st : string;
    fname : str12;
    num : string;
    long : longint;
begin
```



```
    with files[index] do
      begin
        if desig = 255 then fname := '[ DELETED  ]' else
           fname := convert(name);
        long[0] := losize;
        long[1] := hisize;
        case attr of
          chr(8),chr(40)  : num := '  <VOL>';
          chr(16),chr(48) : num := '  <DIR>';
          otherwise num := format_num(long,7);
        end;
        ftime := filetime(time);
        fdate := filedate(date);
        st := concat(fname,'  ',fdate,'  ',ftime,'  ',num);
      end;
  show := st;
end;

function executable(var filename : str12) : boolean;
begin
  executable := ( (pos('.EXE',filename) > 0) or (pos('.COM',filename) > 0) or
                  (pos('.BAT',filename) > 0) );
end;

procedure view_dir(var files : file_array; win : integer; flag : boolean);
var keystat,ascii,code,scan,ytop,ymax : integer;
    name,s4,mask : string[14];
    key : char;
    ft : boolean;
    command,filedat,oldname,newname : string;
    label 88,99;

begin
  ytop := beg_y[win];
  ymax := max_y[win];
  if copy_flag then
    begin
      copy_flag := false;
      flag := true;
    end;
  88: scan := 0;
  code := chdir(dir[win]);
  top_line(ytop,dir[win]);
  if not flag then goto 99;
  scroll_it(ytop+1,8,0);
  mask := '*.*';
  get_files(mask,files,total[win]);
  if total[win] = 0 then goto 99;
  sort_files(files,total[win]);
  index[win] := 0;
  count[win] := 0;
  x[win] := 3;
  y[win] := ytop;
  last_page[win] := (total[win] div 8) + 1;
  if total[win] mod 8 = 0 then last_page[win] := pred(last_page[win]);
  curr_page[win] := 1;
  if (index[win] < total[win]) then
    loop
      count[win] := succ(count[win]);
      index[win] := succ(index[win]);
      y[win] := succ(y[win]);      filedat := show(index[win],files);
      screenwrite(x[win],y[win],main_color,filedat);
      if  (count[win] > 7) or (index[win] >= total[win]) or (total[win] =
0) then
        begin
          y[win] := ytop + 1;
          index[win] := (curr_page[win] * 8) - 7;
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Tom Rettig's Library™

Prewritten Solutions to Programming Problems

Clipper Edition

Advanced Extended Library

- Convenient .LIB library file to link
- Separate object files so only the code actually used is added to your program
- Optimized for speed and code size
- Requires Clipper, Winter '85 or later

dBASE III PLUS Edition

Advanced Programmer's Library

- Interface to C/assembler is fully compatible with Clipper's Extend system
- Functions are CALL procedures
- Requires dBASE III PLUS, any version, and 128K of additional memory

Each edition comes complete with...

- Entire source code, nothing withheld. Cleanly written, liberally commented, easy to modify, helpful for learning.
- Support by phone and electronic mail
- No royalties or copy protection
- Documentation inserts for dBASE or Clipper manual. Accurate, complete, easy to use, page-per-command.
- Handy plastic reference card
- Full money-back guarantee

Over 175 functions; 55% written in C, 35% in Assembler, 10% in dBASE

$99.95 per edition at dBASE/Clipper dealers or direct

II/III Tom Rettig Associates
Excellence in dBASE since 1982

9300 Wilshire Boulevard, Suite 470
Beverly Hills, CA 90212-3237, U.S.A.
(213) 272-3784 ▪ Telex: 4996426 RETTIG
Source: BCR480 ▪ CompuServe: 75066,352

dBASE and dBASE III PLUS are Ashton-Tate trademarks, Clipper is a Nantucket trademark

Call or write for free product information

Number 117 on the Reader Service Card

Minimize Turbo Pascal compile time.

Dramatically reduce debugging time.

Tmark allows Turbo Pascal to continue compiling without returning to line 1.

Tmark is a utility that fixes an annoying limitation of Turbo Pascal. Normally, every time Turbo finds an error while compiling, it must start compiling again from the very beginning. Tmark changes that. It allows Turbo to resume compiling from where you fixed the error, not back at line 1.

Tmark runs Turbo Pascal as a child process. This allows Tmark to monitor Turbo as it compiles your program. When Tmark detects that Turbo is about to issue an error message, it saves, on disk, an image of the memory and registers used by Turbo Pascal up to that point. Once you have fixed the error, Tmark will restore Turbo to its state before the error was found. Turbo will then resume compiling—from the fix.

Tmark will also save up to 7 images at designated lines in a program. When Turbo encounters a line which begins with {TMARK}, Tmark will save an image. You can resume compilation at any of these lines.

Don't lose another minute of compiling time, order Tmark now!

- Not memory resident
- Does not take over any software interrupts. If Turbo Pascal runs on your system, so will Tmark.
- Not copy protected

Tmark™

Only $80

Visa/MC accepted. Add $2 for shipping in U.S., $5 overseas

Tangent Designs, Inc.

P. O. Box 896
Lake Forest, IL 60045
(800) 356-2750 *Outside Illinois*
(312) 295-0030

Turbo Pascal is a registered trademark of Borland International. Supports Turbo versions 3.01 and above, with 8087 or BCD.

Number 126 on the Reader Service Card

PRODUCED 2024 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
99: repeat
      curr_page[win] := (index[win] div 8);
      if index[win] mod 8 <> 0 then
        curr_page[win] := succ(curr_page[win]);
      filedat := show(index[win],files);
      if total[win] = 0 then
        begin
          filedat := 'No files present';
          y[win] := ytop + 1;
        end;
      fx(length(filedat)+1,curs_color,x[win],
              y[win],main_color,filedat);
      repeat until keycode(keystat,ascii,scan);
      if scan = del then
        begin
          purge(files[index[win]].name);
          files[index[win]].desig := 255;
        end;
      if scan = ins then
        begin
          oldname := files[index[win]].name;
          newname := user_entry('Enter new file name');
          rename_file(oldname,newname);
          files[index[win]].name := newname;
        end;
      if (ascii = 0) and (scan = ctrl_home) then
        begin
          dir[win] := root_dir;
          flag := true;
          goto 88;
        end;
      if (ascii = 0) and (scan = ctrl_end) then
        begin
          dir[win] := default_dir;
          flag := true;
          goto 88;
        end;
      if (ascii = 0) and (scan = ctrl_pgup) then
        begin
          s4 := '..';
          code := chdir(s4);
          getdir(dir[win]);
          flag := true;
          goto 88;
        end;
      if total[win] = 0 then return;
      if scan = 46 then copy_file(index[win],win,files);  {'c'}
      if scan = 32 then                                   {'d}
        begin                drive_menu;
          getdir(dir[win]);
          flag := true;
          goto 88;
        end;
      if (scan = retkey) and (files[index[win]].attr = chr(16)) then
        begin
          flag := true;
          if dir[win][length(dir[win])] <> '\' then
          dir[win] := concat(dir[win],"\",files[index[win]].name)else
            dir[win] := concat(dir[win],files[index[win]].name);
          goto 88;
        end;
      if (scan = retkey) and executable(files[index[win]].name) then
        begin
          command := files[index[win]].name;
          execute(command);
          scan := 0;
        end;
```

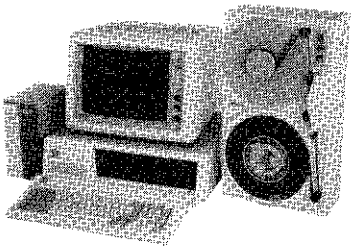


```
if scan = esc then terminate;
if scan = tab then
  begin
    screenwrite(x[win],y[win],main_color,filedat);
    return;
  end;
if (scan in [home,down,up,pgdn,pgup,endkey]) then
  begin
     screenwrite(x[win],y[win],main_color,filedat);
     case scan of
      home : if curr_page[win] > 1 then
               begin
                 index[win] := 0;
                 curr_page[win] := 1;
               end else scan := 0;
    endkey : if curr_page[win] < last_page[win] then
               begin
                 curr_page[win] := last_page[win];
                 index[win] := (last_page[win]  * 8) - 8;
                 scroll_it(ytop+1,8,0);
               end else scan := 0;
      down : begin
               if index[win] = total[win] then
                 begin
                   index[win] :=
                     index[win] - (y[win] - ytop) + 1;
                   y[win] := ytop + 1;
                 end else
               if index[win] + 1 <= total[win] then
                 begin
                   index[win] := succ(index[win]);
                   if y[win] + 1 <= ymax then
                     y[win] := succ(y[win]) else
                       scroll_it(ytop+1,1,0);
                 end;
             end;
        up : begin
               if index[win] = 1 then
                 begin
                   if total[win] > 8 then
                     begin
                       y[win] := ymax;
                       index[win] := index[win] + 7;
                     end
                   else
                     begin
                       y[win] := ytop + total[win];
                       index[win] := total[win];
                     end
                 end
               else if index[win] - 1 >= 0 then
                 begin
                   index[win] := pred(index[win]);
                   if y[win] - 1 >= ytop + 1
                     then y[win] := pred(y[win])
                      else scroll_it(ytop+1,1,1);
                 end;
             end;
      pgup : if curr_page[win] > 1 then
               begin
                 curr_page[win] := pred(curr_page[win]);
                 index[win] := curr_page[win] * 8 - 8;
               end
             else index[win] := 0
      pgdn : if curr_page[win] <= last_page[win] then
               begin
                 if curr_page[win] < last_page[win] then
```


PRODUCED 2050 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

9-Track Tape Subsystem

for the IBM PC/XT/AT
XENIX or
MS-DOS.



The solution to your micro/mainframe communications problem is available today!

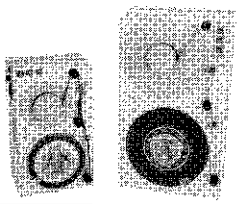
Qualstar's new ½ inch 9-track MINISTREAMER™ brings full ANSI data interchange capability to the PC. Now you can exchange data files with virtually any other computer using 9-track tape.

Available in both 7" and 10½" versions, the MINISTREAMER weighs in at only 27 pounds and uses less desk space than an ordinary sheet of paper, yet provides full 1600/3200 BPI capability at an affordable price. Up to 134 megabytes of data (depending on format) can be stored on a standard 10½" reel of tape, thus making the MINISTREAMER a highly-reliable answer to your backup requirements as well.

Tape subsystem includes tape drive, coupler card, cables, dust-cover and MS-DOS or XENIX compatible software. Prices start at $2,995.

386 READY!

Discover the many advantages 9-track tape has over other Micro/Mainframe links.



Call us today!

QUALSTAR®

9621 Irondale Avenue,
Chatsworth, CA 91311
Telephone: (818) 882-5822

Number **97** on the Reader Service Card

Software Tools

For Programmers & Non-Programmers

Get 'State of the Art' performance and save valuable time with these high quality utilities!

Opt-Tech Sort™

Opt-Tech Sort is a high performance Sort/Merge/Select utility. It can read, sort and write a file faster than most programs can even read the data. Example: 1,000 records of 80 bytes can be read, sorted and a new file written in less than 10 seconds (IBM XT). Opt-Tech Sort can be used as a stand-alone program or called as a subroutine to over 25 different programming languages.

All the sorting, record selection and reformatting facilities you need are included. A partial list of features includes: The ability to process files of any size. Numerous filetypes are supported including Sequential, Random, Delimited, Btrieve, dBASE II & III and many others. Up to 10 key fields can be specified (ascending or descending order). Over 16 different types of data supported. Powerful record selection capability allows you to specify which records are to be included on your output. Record reformatting allows you to change the structure of your output record and to output special fields such as record numbers for use as indexes.

MS-DOS $149. ★ NEW ★ Xenix $249.

★ NEW ★ VERSION On-Line Help™ ★ NEW ★ VERSION

On-Line Help allows you to easily add "Help Windows" to all your programs. On-Line Help is actually two help packages in one. You get BOTH Resident (pop-up) and Callable Help Systems.

The resident version allows you to add help to any system. Your Help System is activated when the "Hot Keys" that you specify are pressed. You can then chain between help windows in any manner you desire.

The callable version allows you to easily display help windows from your programs. A simple call to the help system makes the window appear. The original screen is automatically restored when the help window is cleared. On-Line Help is callable from over 20 different languages.

You have full control over the help window content, size, color and location.

MS-DOS $149. **Demo $10.** (apply toward purchase).

Scroll & Recall™

Scroll & Recall is a resident screen and keyboard enhancement. It allows you to conveniently scroll back through data that has gone off the top of your display screen. Up to 27 screens of data can be recalled or written to a disk file (great for documenting systems operations). Also allows you to easily recall and edit your previously entered DOS commands without retyping.

Scroll & Recall is very easy to use. It's a resident utility that's always there when you need it. MS-DOS $69.

Visa, M/C, AMEX, Check, Money Order, COD or Purchase Orders accepted.

To order or to receive additional information just call and receive immediate highly qualified attention!

Opt-Tech Data Processing
P.O. Box 678 — Zephyr Cove, NV 89448
(702) 588-3737

Number **16** on the Reader Service Card

$99 BUSINESS BBS

The Business Board System (BBS) provides a 24 hour dial-up business information center using 300:1200:2400 baud modems. The user configurable menus allow viewing (product) info easily. Similarly, information can be collected from a caller (product order) or file upload/download (product delivery). Integrated data base & remote PC operation programs. With source code $249.

$99 COMMX C

Reliable communications library for C programmers that works with ALL Hayes compatible internal & external modems! Its been tested in a 386 at 16mhz. Both small & large memory models are supported. Written in ASM for small code size 1k:1k isr buffer. Timer fcns provide proper operation at any speed CPU and modem. Includes C programs for testing. Complete ASM source code $249.

$59 C DATA ENCRYPTION

US government standard per FIPS PUB46 - DES in Microsoft "C". Electronic Code Book, Cipher Block Chain, and Cipher Feed Back. Includes compression for faster transmission & less storage. Includes telecom formatting allowing spreadsheets or ANY file to be sent via information services (CompuServe, Source, TLX/TWX). Simple menu driven operations. Complete "C" source code $249.

HAWKEYE GRAFIX Inc
BOX 1400, OLDSMAR
FLORIDA 33557
DIAL 813-855-5846

Number **57** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

PJ MICRONOTE

Shareware Professionals Form Association

by
John J. Newlin

A group of dedicated Shareware programmers have formed the Association of Shareware Professionals (ASP) in order to strengthen the future of Shareware (user supported software) as an alternative to commercial software. Marshall Magee (Automenu) and Bob Wallace (PC-Write), two prominent and successful Shareware authors, decided that closer cooperation among Shareware authors could benefit the Shareware concept of marketing by educating the consumer public and distributors of Shareware. Nelson Ford (DiskCat), a Shareware author as well as a public domain and Shareware distributor, had similar feelings and organized a conference in February of this year to discuss the situation.

The result of that conference is the ASP. Marshall Magee approached the CompuServe IBMNET System Operators (Sysops) and requested that the ASP be granted permission to use a dedicated area of the IBM PC forum. The Sysops consented by creating a special data library for the ASP in the IBMJR section of IBMNET. Jim Button (PC-File) posted a message in the CompuServe IBM software forum inviting Shareware authors to apply for membership. The criterion for ASP membership is basically that the applicant be a Shareware professional as evidenced by his or her marketing of a quality, non-trivial program as Shareware. At this writing, the ASP is sixty members strong and growing. Barry Simon (CtrlAlt), Robert Tolz (PAL), and Howard Benner (zapcis) led the effort that resulted in membership adoption of bylaws. Jim Button was persuaded to act as temporary chairman until the board of directors was elected.

In what is very likely an historic first, the members of the ASP drafted and adopted its bylaws and elected a board of directors entirely through the electronic medium provided by the IBMNET Sysops. The ASP goals are

- Inform users about Shareware programs and about Shareware as a method of distributing and marketing software.
- Encourage broader distribution of Shareware through user groups and disk dealers who agree to identify and explain the nature of Shareware.
- Assist members in marketing their software.
- Provide a forum through which ASP members may communicate, share ideas, and learn from each other.
- Foster a high degree of professionalism among Shareware authors by setting programming, marketing, and support standards for ASP members to follow.

The new board of directors elected by the membership are

Jim Button (PC-File)
Nelson Ford (DiskCat)
Marshall Magee (Automenu)
Neil Rubenking (Pianoman)
Tom Smith (ProComm)
Barry Simon (CtrlAlt)

Shareware programmers interested in joining the ASP may obtain an application form by downloading the file APPLIC.FRM from Data Library 8 (Village Inn) of the CompuServe IBM PC Software SIG. The ASP membership is most grateful to IBMNET Sysops Don Watkins, Conrad Kageyama, Chris Dunford, and archivist Joan Friedman for their assistance and support. ♦

25

Newlin Code, Continued

```
begin
    index[win] := curr_page[win] * 8;
    curr_page[win] := succ(curr_page[win]);
    scroll it(ytop+1,8,0);
  end else scan := 0;
 end;
 end; {of case}
 end; {of if scan in []}
until scan in [home,endkey,pgup,pgdn];
y[win] := ytop;
count[win] := 0;
end; {of if count[win]}
end; {of loop}
end;

procedure initialize;
begin
  window_flag := true;
  getdir(current_dir);
  root_dir := copy(current_dir,1,3);
  draw_box(win_col,win1_beg,46,9);
  draw_box(win_col,win2_beg,46,9);
  draw_box(1,1,78,3);
  dir[1] := current_dir;
  dir[2] := root_dir;
  beg_y[1] := win1_beg;
  beg_y[2] := win2_beg;
  max_y[1] := win1_beg + 8;
  max_y[2] := win2_beg + 8;
end;

begin
  paragraphs := set_mem;
  get_drive_list;
  cls(15);
  save_cursor;
  hide_cursor;
  getdir(default_dir);
  initialize;
  str := ' The Shell Game - by John Newlin ';
  screenwrite(6,1,main_color,str);
  loop
    view_dir(files[1],1,window_flag);
    view_dir(files[2],2,window_flag);
    if window_flag then window_flag := false;
  end;
end.
```


PRODUCED 2005 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

UNLEASH YOUR 80386!

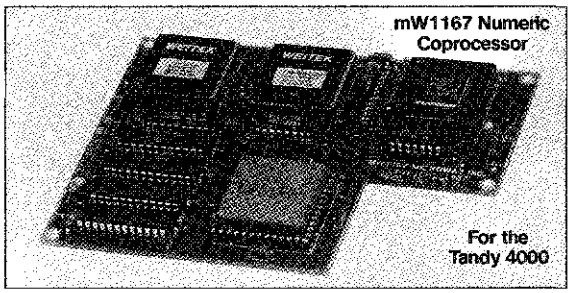
Your 80386-based PC runs at least twice as fast as your old AT. This is good, but not great. The products described below will unleash the true potential of your 80386, giving you 4 to 16 times the power of your old AT. These new MicroWay products include a family of 80386 native code compilers and the mW1167 numeric coprocessor.

Examples of the increases in capacity and performance include:

- Programs compiled with MicroWay NDP Fortran-386 execute 2 to 8 times faster than those compiled with existing 16-bit Fortrans. NDP Fortran-386 can also address up to 4 gigabytes of memory instead of the standard 640 kbytes. MicroWay's NDP compilers and the programs they generate run on MS-DOS or Unix V.
- NDP Fortran-386 generates code for the 80287, 80387 or MicroWay's mW1167. The mW1167 has a floating point throughput exceeding 2.5 megaflops, which is 4 to 5 times the throughput of an 80387 and is comparable to the speed achieved by the VAX 8600.

Equally important, whichever MicroWay product you choose, you can be assured of the same excellent pre- and post-sales support that has made MicroWay the world leader in PC numerics and high performance PC upgrades. For more information, please call the Technical Support Department at

617-746-7341



MicroWay® 80386 Support

MicroWay 80386 Compilers

NDP Fortran-386 and **NDP C-386** are globally optimizing 80386 native code compilers that support a number of Numeric Data Processors, including the 80287, 80387 and mW1167. They generate mainframe quality optimized code and are syntactically and operationally compatible to the Berkeley 4.2 Unix f77 and PCC compilers. MS-DOS specific extensions have been added where necessary to make it easy to port programs written with Microsoft C or Fortran and R/M Fortran.

The compilers are presently available in two formats: Microport Unix 5.3 or MS-DOS as extended by the Phar Lap Tools. MicroWay will port them to other 80386 operating systems such as OS/2 as the need arises and as 80386 versions become available.

The key to addressing more than 640 kbytes is the use of 32-bit integers to address arrays. NDP Fortran-386 generates 32-bit code which executes 3 to 8 times faster than the current generation of 16-bit compilers. There are three elements each of which contributes a factor of 2 to this speed increase: very efficient use of 80386 registers to store 32-bit entities, the use of inline 32-bit arithmetic instead of library calls, and a doubling in the effective utilization of the system data bus.

An example of the benefit of excellent code is a 32-bit matrix multiply. In this benchmark an NDP Fortran-386 program is run against the same program compiled with a 16-bit Fortran. Both programs were run on the same 80386 system. However, the 32-bit code ran 7.5 times faster than the 16-bit code, and 58.5 times faster than the 16-bit code executing on an IBM PC.

NDP FORTRAN-386™ $595
NDP C-386™ $595

MicroWay Numerics

The **mW1167™** is a MicroWay designed high speed numeric coprocessor that works with the 80386. It plugs into a 121 pin "Weitek" socket that is actually a super set of the 80387. This socket is available on a number of motherboards and accelerators including the AT&T 6386, Tandy 4000 and MicroWay Number Smasher 386 (Jan. '88). It combines the 64-bit Weitek 1163/64 floating point multiplier/adder with a Weitek/Intel designed "glue chip". The mW1167™ runs at 3.6 MegaWhetstones (compiled with NDP Fortran-386) which is a factor of 16 faster than an AT and 3 to 5 times faster than an 80387 .....$1495

Monoputer™ – The INMOS T800-20 Transputer is a 32-bit computer on a chip that features a built-in floating point coprocessor. The T800 can be used to build arbitrarily large parallel processing machines. The Monoputer comes with either the 20 MHz T800 or the T414 (a T800 without the NDP) and includes 2 megabytes of processor memory. Four or more Transputers can be easily linked together to form a Quadputer. A single T800 is comparable in speed with an mW1167-equipped 80386. The compilers to drive one or more Monoputers include Occam, C, Fortran, Pascal and Prolog.

Monoputer T414-20[1] $1495
Monoputer T800-20[1] $1995
Biputer™ T800/T414[2] $4995
Quadputer™ T414-20[2] $6995

[1] Includes Occam [2] Includes TDS

80287 ACCELERATORS

287Turbo-10 .....$450
287Turbo-12 .....$550
287TurboPlus-12 .....$629

80386 Multi-User Solutions

AT8™ – This intelligent serial controller is designed to handle 8 users (16 with two boards) in a Xenix or Unix environment with as little as 3% degradation in speed. It has been tested and approved by Compaq, Intel, NCR, Zenith, and the Department of Defense for use in high performance 80286 and 80386 Xenix or Unix based multi-user systems $1299

MicroPort Unix 5.3 is a port of the new Unix 5.3 to the 80386. MicroWay NDP-386 compilers currently run on this version of UNIX.
MicroPort Unix 5.3 .....from $399

PC-MOS-386™ is an 80386 operating environment that turns an AT with an AT8 into an MS-DOS multi-user system. The system makes it possible to run applications such as Lotus 1-2-3 on terminals. The operating system also has a Phar Lap compatibility mode that runs programs developed with the Phar Lap versions of MicroWay's compilers .....from $199

Phar Lap™ created the first tools that make it possible to develop 80386 applications which run under MS-DOS yet take advantage of the full power of the 80386. These include an 80386 monitor/loader that runs the 80386 in protected linear address mode, an assembler, linker and debugger. These tools are required for the MS-DOS version of the MicroWay NDP Compilers.
Phar Lap Tools.....$399

MATH COPROCESSORS

80387-16 16 MHz.....$495
80287-10 10 MHz.....$349
80287-8 8 MHz $259
80287-6 6 MHz $179
8087-2 8 MHz $154
8087 5 MHz $99

MicroWay

The World Leader in PC Numerics

P.O. Box 79, Kingston, Mass. 02364 USA (617) 746-7341
32 High St., Kingston-Upon-Thames, U.K., 01-541-5466

Number **104** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Turbo Pascal's Renaissance

by
Tom Swan

Turbo Pascal 3.0 is alive and well and living in 600,000 Flip 'N' Files across the land. Lately, though, the former new kid on the block has been feeling its age. Some of the spunk is gone. Its wrinkles show a little deeper. Now an ancient five-year-old, Turbo Pascal ain't no young chicken no more. (Excuse the exquisite grammar, please.)

Enter Turbo Pascal 4.0, no howling baby in rebirth, but a sophisticated grownup, newly outfitted with an integrated editor/compiler package, Turbo-C style menus and windows interface, improved compile and runtime performance, command-line and integrated compiler versions on disk, a Make facility, "smart" linker built in, math coprocessor and emulation support, separate compilation, new data types, IEEE real numbers, short-circuit Booleans, re-entrant runtime library, nested include files, better assembly language linkage, an improved memory map, FAR procedures and functions, and to quote merchants everywhere, "much, much more."

The following is a preview of features in the all new Turbo Pascal 4.0, scheduled for release late this year. To borrow *BYTE*'s famous disclaimer, this "is not a review." All of this information comes from spending many hours with a very early Beta copy of the compiler and from reading between the lines in the scanty documentation provided to me. So if any of this proves wrong, my apologies.

Double Trouble

On disk are two versions of the compiler: integrated and command-line. The integrated version (TP.EXE) is a self-contained development package, including editor, compiler, and linker. The command-line version (TPC.EXE) runs directly from DOS and will appeal to those who prefer using a separate editor. To compile a program with TPC, you type something like this:

```
TPC -M MYPROG
```

The -M is a compiler command to compile (Make) out-of-date modules, similar in purpose to the Make program included with most C compilers. Other options select various features: -F finds runtime errors, -I specifies a directory for include files, and so forth. The compiler translates MYPROG.PAS directly to MYPROG.EXE. There are no intermediate assembly or link steps to perform.

The integrated package is more like the Turbo Pascal of old although the interface is sand-blasted fresh. (The built-in editor uses the same WordStar control-key sequences as 3.0's.) With this version of the compiler, you select commands through pull-down menus and "hot keys." A lot of thought has gone into the interface. For example, instead of typing file names, you position a highlighted bar over a directory entry and press Return. Pressing F2 at any time saves the current work file. Pressing Alt-R compiles and runs the current program after which you return to the editor by pressing Return. This means no more flipping and flopping from the editor to the Main Menu page. All other commands and features are easily accessible through similar well conceived Function-, Alt-, and Control-key combinations.

For most purposes, both the command-line and integrated packages are functionally identical. Both are 100% source code compatible. The command-line version occupies less memory, though, and might be able to handle larger jobs with elephantine symbol tables that cannot fit in memory along with the integrated package. Most programmers will never need to use command-line Turbo Pascal, but even so, it's certainly nice to have the option included at no extra charge.

If 3.0 was fast, 4.0 is ballistic!

I tested the integrated compiler on a Magnavox Multimode display with a Vega Deluxe card. Turbo Pascal's display colors make windows, menus, and error messages stand out clearly. Eagle-eye programmers with EGA cards will appreciate an option that shows up to 40 lines at once. The integrated compiler is a breeze to use. I never needed to consult a manual, which is lucky because I didn't have one anyway.

Code Generation

Turbo Pascal 4.0 flies like a rocket, compiling two or three times faster than 3.0. If 3.0 was fast, 4.0 is ballistic! My informal tests show runtime speeds anywhere from five to 35 percent better, too. I suspect some of this improvement comes from faster display writes—programs that spend the most time

PRODUCED 2005 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

writing to the display show the best improvements. This isn't the only factor, though. The compiler optimizes the code it generates, for example, using short jump instructions whenever possible.

As in 3.0, the new integrated system can compile either to disk or to memory. This lets you compile and run a program from inside Turbo or compile to disk, postponing execution until later. A menu command lets you temporarily return to DOS and then type EXIT to get back to Turbo Pascal. You can use this command to test-drive a program from DOS—or to use any other DOS command—and then quickly return to the editor to continue working.

Both the integrated and command-line compilers can create EXE files from which Turbo's "smart" linker strips dead wood, drastically reducing code file size on disk. Previously, every disk code file (except CHN files) had the entire runtime library attached. This is no longer true. For example, one of my test EXE files occupies only 6,912 bytes when 4.0 compiles the program. Compiling that same test with 3.0 creates a 12,243-byte COM file. (It's not clear whether you will still be able to create COM files with the new compiler, but if not, this will be no great loss.)

One unresolved code-generation issue concerns overlays and chaining. The compiler I tested doesn't have either of these features, but I'm told that overlays might go back in. Some people have criticized overlays as inefficient although it's my belief that, when properly used, overlays do not significantly degrade performance. The key phrase is "properly used." Abuse overlays and your program will stagger like a Tyrannosaurus in a tar pit.

If overlays and chaining do not make it into the production compiler, though, to segment large programs, you'll have to employ the DOS Exec function call, supplied in a library unit on disk. (Units are separately compiled modules—more about them later.) My vote is to keep overlays—and my guess is that the 4.0 compiler will eventually, if not initially, have them. The absence of overlays is a gaping hole begging critics to pour in their vile venom. Surely Borland realizes this?

Table: Turbo Pascal 4.0 Numeric Data Types

Data type	Bytes	Range
Byte	1	255
ShortInt	1	128..127
Word	2	0..65,535
Integer	2	–32,768..32,767
LongInt	4	–2,147,483,648..2,147,483,647
Single	4	3.4E–38..3.4E+38
Real*	6	1E–38..1E+38
Double	8	.7E–308..1.7+308
Extended	10	1.1E–4932..1.1E+4932
Comp	8	9.2E–18..9.2E+18 (approx.)

* From Turbo Pascal 3.0

Data Types

The usual data types—CHAR, Boolean, sets, pointers, and enumerated values—remain unchanged. Type String now defaults to 255 characters if you do not specify a length in brackets as in S:STRING[64].

The table shows 4.0's ten numeric data types. Byte and Integer types are identical in 3.0 and 4.0. Finally, we have signed and unsigned bytes and words—a long overdue addition! The LongInt (long integer) type increases the usual integer ranges and plays an important role in disk file handling. Many file routines such as BlockRead and Seek now use and return LongInt instead of Integer values. For this reason, Real-number functions LongSeek and LongFileSize are no longer recognized, nor are they needed.

New IEEE real numbers modernize the compiler's calculating abilities. Single precision, 4-byte reals offer the best calculation speeds. Double precision 8-byte reals improve the accuracy of the default REAL 6-byte format. The 10-byte extended type offers an extreme range and the highest precision.

The Comp type requires explanation. According to the Macintosh Turbo Pascal manual, Comp is "a counting type that counts things exactly. . . . The sum, difference, or product of any two Comp values can be calculated exactly if the magnitude of the result does not exceed 2^{63-1} (9,223,372,036,854,775,807)."

I'm not treading thin air by quoting the Macintosh manual, by the way. IBM PC Turbo Pascal 4.0 mirrors many features in the Macintosh compiler. In fact, Turbo Pascal Beta testers are given portions of the Mac manual as reference material!

A compiler switch lets you choose between software emulation or hardware coprocessor. The way things stand now, the Single, Double, Extended, and Comp data types require a coprocessor.

Syntax

Short circuit Boolean expressions allow programming such as the following method for removing leading blanks in a string:

```
CONST    BLANK=' ';
VAR      s:STRING[80];

ReadLn  ( s );
WHILE   ( Length(s) > 0 ) AND
        ( s[1] = BLANK  ) DO
   Delete( s, 1, 1 );
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Previously, the Boolean AND expression caused a faulty index operation on zero-length strings. (Indexing s[1] when the length of the string is zero is clearly an error although in this example it does no real harm.) Turbo Pascal's short circuit Booleans eliminate the problem by guaranteeing that evaluation stops as soon as the result of the entire expression is known. In this case, when the string length is zero, the expression (s[1]=BLANK) is skipped.

You can defeat short circuit Booleans by throwing a special compiler switch. This lets masochistic programmers continue to write Boolean expressions and loops the hard way.

A new operator (@) returns the address of a variable, procedure, or function. The following test program demonstrates how to use @:

```
PROGRAM Test;
VAR   p :  Pointer;
      a : ARRAY[ 1..10 ] OF INTEGER;
      i : INTEGER;
BEGIN
   FOR i := 1 TO 10 DO
      a[i] := i;
   FOR i := 1 TO 10 DO
   BEGIN
      p := @a[i];
      Writeln( INTEGER(p^) )
   END
END.
```

Pointer is a new generic type, compatible with pointers of any other type. This example first fills a 10-integer array and then uses the @ operator to assign to pointer p the address of each arrayed value. The Writeln statement uses type casting (formerly called retyping) to inform the compiler that pointer p addresses an integer.

Files

New file variables are well documented. It's now possible to design your own text file I/O device drivers. To do this, you write procedures and stuff their addresses into a file variable. Turbo Pascal then calls your routines for Assign, Reset, Rewrite, Append, Close, Flush, and all I/O statements such as Write, Readln, EOF, plus others that refer to the file.

IoResult, which returns the error code of the most recent I/O operation when the compiler switch {$i-} is in effect, now returns DOS error codes rather than Turbo's own numberings. Also, I/O routines like Reset and Write no longer check Turbo's internal error variable. In Version 3.0, any I/O error prevents further I/O until IoResult clears the internal variable. With 3.0, if you write

```
Reset( f );
Close( f );
Writeln( IoResult );
```

and if Reset fails, the IoResult value holds the correct error code because the Close statement does not execute if Reset's IoResult is not zero. With the new compiler, Close ignores the previous IoResult and issues its own error. A Reset fault in this example, therefore, might be missed or misinterpreted.

Separate Compilation

Being able to compile programs in pieces and then link those pieces together to produce a finished code file are among the most welcome of Turbo Pascal's new features. You now can develop precompiled libraries of common routines for many programs to share.

Turbo Pascal 4.0 accomplishes separate compilation through UCSD Pascal style units. In text form, a unit contains an interface section that defines the items in the unit and an implementation section that contains the actual coding to go along with the definitions.

Compiling a unit produces a TPU (Turbo Pascal Unit) file on disk with the interface text in a binary form that the compiler quickly loads into memory. When a program _USES_ a unit, the compiler reads the TPU file, makes the unit's declarations available to the program, and links the unit code to the finished result. A supplied librarian program stores multiple units in TPL (Turbo Pascal Library) files for faster loading.

Routines declared in the unit interface are addressed via FAR calls. Routines declared local to the unit are NEAR as are all routines in the main program. (You can optionally force the compiler to generate FAR procedures, too.) Because the main program code and each unit go into separate memory segments, the compiler can now build programs larger than the old 64K code size limit.

Even if you don't write your own units, you'll still use them heavily. Nearly all standard routines such as ClrScr and TextColor are now contained in several supplied units. When the integrated compiler runs, it loads these units from a library file, making the standard routines instantly available to programs. The definitions for file devices such as LST and CON are stored in a unit, too. Other units provide procedures and functions similar to those in Version 3.0. Using these units helps compile older programs with statements that 4.0 normally rejects.

Machine Language

There are now three ways to add machine language to Turbo Pascal programs. The original InLine statement survives, apparently with pretty much the same personality. The old External declaration is gone—and good riddance to it. The new methods are far superior.

Turbo Pascal 4.0 accomplishes separate compilation through UCSD Pascal style units.

You can now assemble separate modules to .OBJ files with the Microsoft Macro Assembler 4.0 and later versions and then link the result into Pascal programs as external procedures and functions. In the source text, you declare such routines this way:

```
PROCEDURE Flop( VAR a, b : INTEGER ); External;
FUNCTION Add10( VAR q : REAL ) : REAL; External;
```

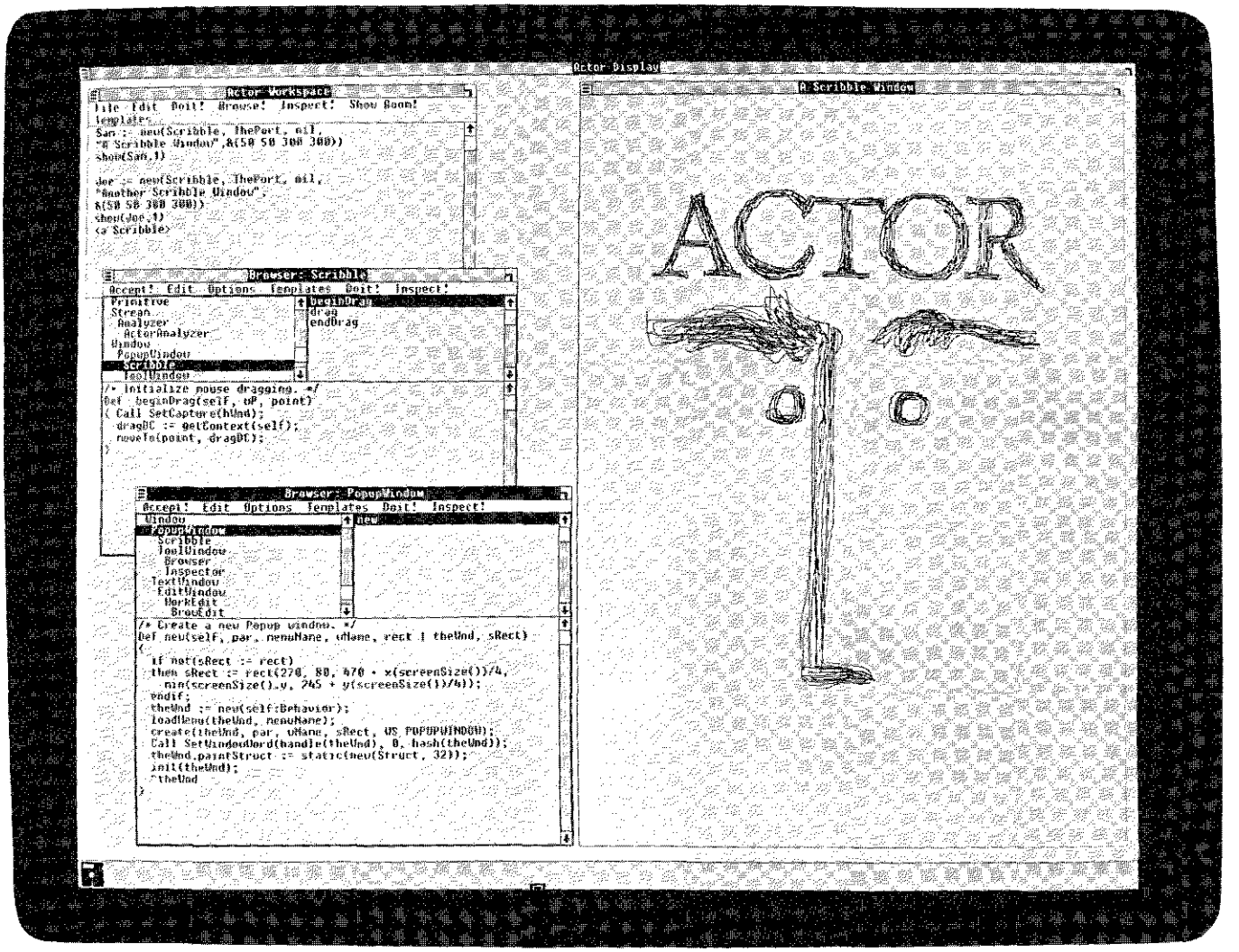
Another new way to add machine code to programs comes from the Macintosh compiler. Similar to the InLine statement, this new method takes the form of a procedure or function declaration such as

```
PROCEDURE MyMacro; InLine( $90/$90/$90 );
```

There are no bodies to these InLine declarations, which operate more like assembly language macros than Pascal routines. In this example, wherever you write MyMacro, the compiler inserts the bytes from the InLine expression. It inserts nothing except those bytes—no entry or exit overhead, no stack adjustment, no CALL or RET instructions. The InLine code is

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

HOW TO WRITE A WINDOWS APPLICATION IN TEN MINUTES.



Actor™ is a new language that combines Microsoft® Windows with object-oriented programming. This means you can produce mouse and window applications very quickly.

For example, we created a simple "paint" program, and used it to draw the Actor logo you see on the screen. The whole program only took ten lines and ten minutes. Part of it is in the middle window on the left.

Above, you see the commands that initialized the paint window and made it appear on the screen. Below, some code that's built into Actor, specifying window behavior. Through a process known as "inheritance," it's called into play automatically.

Try programming in this new way, and you'll never go back.

Find out about Actor. Call The Whitewater Group®, (312) 491-2370.

Technology Innovation Center
906 University Place, Evanston, IL 60201

MICROSOFT® WINDOWS

Number 73 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

not a subroutine—it's a direct injection into the program's blood stream. Continuing the previous example, if you write

```
BEGIN
   Write( 'This is a' );
   MyMacro;
   Write( 'test.' )
END.
```

the compiler inserts three NOP instructions ($90) into the code between the two Writes. No doubt, clever programmers are going to figure out marvelous uses for these new ways to write completely unportable, machine dependent, Pascal programs.

Memory Management

A separate data segment limited to 64K and addressed by DS now contains typed constants (which I prefer to call *variable constants* because all constants have associated data types, no?). These are no longer stored in the code segment as they are in 3.0.

Stack size is now adjustable up to one full 64K segment. The default stack is 8K, large enough for most small- to medium-size programs.

The heap pointer is a 4-byte normalized value, allowing dynamic addressing of free RAM. Normalized pointers ensure that every byte in memory has a unique address, solving the 8086-family idiosyncracy that allows different segment and offset values to address the same locations. (Turbo Pascal 3.0 also uses a normalized heap pointer.)

The new compiler supports pointer arithmetic, letting you increment and decrement pointers in amounts relative to the items the pointers address. (This feature did not work in the Beta compiler I tested—I'm relying on promises made in the documentation.) Adding 5 to a pointer, for instance, advances a pointer 5 items beyond its present position—not 5 bytes. If a pointer addresses 2-byte integers, subtracting 20 moves the pointer back 40 bytes. This kind of pointer arithmetic, familiar to C programmers, lets you change the size of variables addressed by pointers without also changing the pointer arithmetic expressions. All of this was difficult to accomplish in 3.0.

Previously, you could allocate a minimum of 8 bytes on the heap. Although 4.0 can allocate single bytes of heap space, because disposing single bytes can eventually overflow the disposed memory block free list, this new feature has questionable value.

You can now install your own memory-error handler to trap out-of-heap-space conditions. You might use this feature to add a garbage collector to compact the heap, dump a heap snapshot to the printer for debugging memory errors, and so on.

MemAvail and MaxAvail now return values in bytes instead of 16-byte paragraphs as before. This change will break a lot of source code, and I wish new routines (ByteAvail and MaxBytes or something) had been added instead, leaving the old functions alone. If you use MemAvail and MaxAvail, you'll have some work to do to prevent nasty bugs in programs that will now think they have 16 times more memory than before!

Miscellany

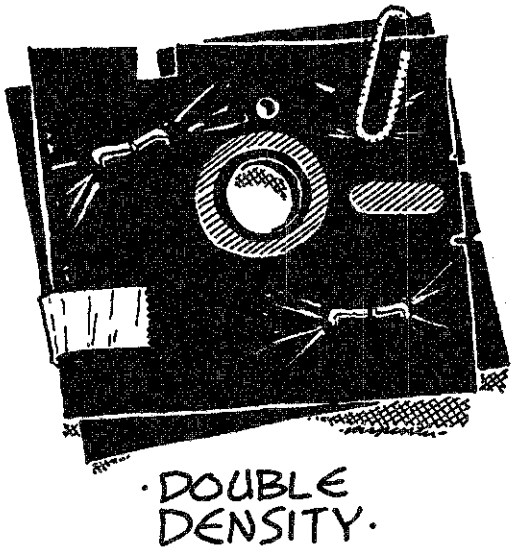
There are several other changes and probably more of which I'm unaware. You can nest include files eight levels deep, letting one include file load another. The compiler can generate map files and debugging information that MASM's SYMDEB can use.

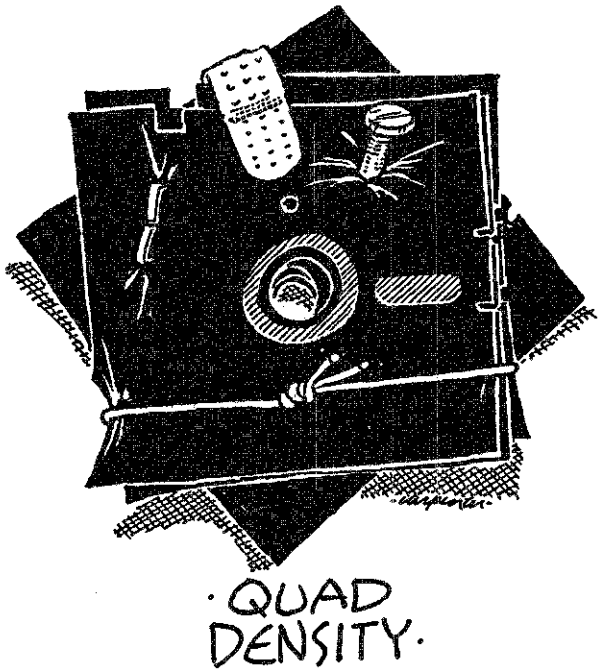
Most of the Turbo Pascal runtime library is interruptable and re-entrant—therefore, you can write interrupt service routines in Pascal. A special keyword (INTERRUPT) is provided for this purpose.

A set of hooks lets you attach your own shutdown and error handler routines to the runtime library. When a program ends or aborts due to an error, your own routines can gain control.

With its many enhancements, Turbo Pascal 4.0 is a renaissance in Pascal programming. Separate compilation, an improved interface, new data types, better assembly language linkage, and other features answer many criticisms of 3.0. If Pascal is your language, even if it's not Turbo, you won't go wrong by choosing this newborn version of Borland's number-one son. ♦

Tom Swan is the author of Mastering Turbo Pascal *(Howard W. Sams),* Mastering Turbo Pascal Files *(also from Sams), and* Programming with Macintosh Turbo Pascal *(John Wiley & Sons). Tom is keeping his disk drives warm revising* Mastering Turbo Pascal *for publication early in 1988.*





PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Create 68K Embedded Systems On ATs

Oregon Software Brings VAX and VME Pascal-2® Cross-Development Tools To MS-DOS Workstations

Get All Your Tools In One Package

The Pascal-2 Cross-Development System gives you price/performance value that beats high-priced workstations. Compare these features and call for further information.

Compiler

- 68000, 68020, and 68881 instruction sets
- ROMable code
- reentrant code
- separate program sections for code and data
- easy access to hardware through fixed memory locations (I/O space)

Assembler-Linker

- VERSAdos compatible output
- easily linked, relocatable S-record format

Concurrent Program Support

- process synchronization primitives for creating a multiprocessing real-time kernel
- interactive post-mortem analyzer
- software-selectable interrupt levels

Stand-Alone Library

- source supplied: users may adapt code as needed
- user programs that run without an operating system or other real-time kernel
- compact code for extremely small ROM programs or limited memory devices
- ability to write device drivers in Pascal or assembly code
- user-customized system services or control programs
- *easily-gathered performance analysis statistics*

Source Management

- a version-control system to track and identify each phase of development
- optional branching to multiple versions from a single source
- automated rebuilding of programs from component modules (a Unix-style Make function)
- a restricted-access system for coordinating revisions

OREGON SOFTWARE

800-367-2202

6915 SW Macadam Avenue, Portland, Oregon 97219 U.S.A.
503-245-2202 FAX 503-245-8449 TWX 910-464-4779

The following are trademarks: Pascal-2, SourceTools, and Oregon Software, Oregon Software, Inc., VAX and VMS Digital Equipment Corporation, UNIX, AT&T Bell Laboratories, Incorporated. MS, and MS-DOS are registered trademarks of Microsoft Corporation.

Number **102** *on the Reader Service Card*

MICROCACHE™ and SPEEDPRINT™

The hottest PC utilities you can get.
Or your money back!

MICROCACHE™ monitors disk use and "learns" to anticipate your data accesses. Then, by satisfying these requests directly from RAM instead of disk, your programs run as if they are SUPERCHARGED.

- 100% Transparent Operation
- Uses any spare RAM available
- EMS/EEMS memory support
- SPEEDPRINT™ buffer option
- Requires NO program changes
- For IBM and compatibles

Introductory Price $89.00

MasterCard VISA

Az-Tech Software, Inc.
305 East Franklin
Richmond, MO 64085

(800) 227-0644 . . . (816) 776-2700

Number **51** on the Reader Service Card

TURBO SOURCE SEARCH

"BBS-by-mail"

2 FREE DISKS then just $2.50 A DISK

TURBO SOURCE SEARCH (TSS) is the largest SIG library of Turbo Pascal public domain/shareware programs.

TSS is in fact a Bulletin Board System (BBS). We have the usual modem-connect BBS (617-545-9131), but also offer a BBS-by-mail service where members may get the files from the library on disks by mail for just $2.50.

TSS members pay a one time fee of $25 and receive a set of catalog disks plus credit for 2 FREE DISKS.

TSS has over 110 disks, 35Mb of compressed files, of software. These files have been placed into public access channels by their authors and collected by TSS so that you can avoid the costs of collecting them yourself.

And since TSS is turning ONE YEAR OLD we are offering a one price special of $250 for full membership PLUS the full library of disks updated through disk 150 which should be added to the library by mid-summer!!! (Current members: $2 per disk, 50+ disk orders.)

For a set of catalog disks send $5, or send $25 to become a full member of TSS, the "BBS-by-mail" to:

TURBO SOURCE SEARCH
P.O. BOX 876, SCITUATE, MA 02066, 617-545-6677
data/BBS: 617-545-9131 (anyone may call!)

VISA, MASTERCARD, COD $5 extra, non-usa orders add $10

Number **35** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Handling Fixed Length Char Arrays

by
William J. Hunt

Although the C language provides special support for string constants (i.e., "**ABC**") and the **printf** and **scanf** functions have special "**%s**" formats, fixed length char arrays padded with blanks are sometimes better suited to the application. For example, when structured data is stored in fixed length records on disk files, the terminating '**\0**' character wastes space. In addition, C programs may have to read existing files with fixed length char array fields. Fortunately, C provides some useful library functions for handling fixed length char arrays and writing additional char array handling functions in C is easy.

In addition to the familiar **strxxx** string functions, recent versions of Lattice and Microsoft C support the **memxxx** functions defined by the ANSI C Draft Standard. These functions use a byte count to define the area rather than the '**\0**' string terminator the **strxxx** functions expect. The function prototypes below define these library functions and list the comparable **strxxx** functions:

```
#include "string.h" /* defines return values, etc. */

void  *memcpy(  void *s1, void *s2, size_t n) ;
void  *memmove( void *s1, void *s2, size_t n) ;
      move n bytes starting at the address s2 to
      the area starting at address s1. memmove copies
      overlapped areas correctly while memcpy
      has no guarantee.
      (strcpy)

int memcmp( const void *s1,const void *s2,size_t n) ;
      compares n bytes starting at s1 and s2 addresses.
      returns  < 0 if object at s1 < object at s2.
               = 0 if object at s1 == object at s2
               > 0 if object at s1 > object at s2
      (strcmp)

void  *memchr(const void *s, int c, size_t n) ;
      locates 1st occurrence in s of the char c.
      returns NULL if c not matched in n chars.
      (strchr)

void  *memset(void *s,int c, size_t n) ;
      copies the value of c (as unsigned char) to
      n bytes starting at address s.
      (no strxxx equivalent.)
```

These function prototypes are taken from the ANSI C draft document and contain some new features not yet fully supported by Lattice or Microsoft C. Neither compiler allows dummy argument names in function prototypes. ANSI C defines the **size_t** data type in the **limits.h** and **stddef.h** header files. Current versions of Lattice C and Microsoft C expect an unsigned data type for the length argument. The **void *** type is a generic pointer to an object in memory—any data type. In the **memcmp** declaration, the word **const** tells the compiler that the data areas to which **s1** and **s2** point will not be altered by **memcmp**.

These **memxxx** functions don't do everything, so we need to write some macros and functions of our own. The **blank_fill** macro below fills a char array of length n with blanks, and the **trim** function truncates a string, removing trailing blanks.

```
#define blank_fill(fa,n)    ( memset((fa),'b',(n)) )

/* trim.c - trim string function */
#include "stdio.h"

int trim(s)                 /* trim non-blanks from string */
 char *s ;
 {
   int n ;

   n = strlen(s) - 1 ;      /* start at end of string */
                            /* stop at for right-most non-blan
                            /* or at beginning of string */
  while( (n >= 0 ) && (s[n] == ' ') )
     { n-- ; }
   n++ ;                    /* new length of string */
   s[n] = '\0' ;            /* mark the end */
   return( n ) ;
 }
```

With these primitive operations, we can easily convert between string format and a fixed length array:

```
                            /* string to fixed array */
blank_fill(fa,n) ;          /* clear array to blanks */
memcpy(fa,s,strlen(s) ) ;   /* copy chars from string */

                            /* fixed array to string  */
memcpy(s,fa,n) ;            /* copy entire array      */
s[n] = '\0' ;               /* terminate string       */
trim(s) ;                   /* remove trailing blanks */
```

Other languages such as BASIC and Pascal provide a string data type with a 2-byte integer string length before the char array itself. Such a format could be declared in C as

```
typedef struct              /* Pascal format string */
 { int slen ;               /* string length */
   char sa[ MAX_SIZE ] ;    /* store chars here */
 } PSTRING ;
```

The **memxxx** functions would also be useful for handling strings in this format. Of course, BASIC and some Pascals provide string assignment with dynamic allocation of strings and automatic garbage collection. Such features require lots of support and potential overhead and so are not provided by the C library. That is a subject for another article. ♦

William J. Hunt writes the Lattice and Microsoft C Users' Group *newsletter, where this material first appeared, and is also the author of* The C Toolbox, *published by Addison-Wesley. Readers may contact Bill at the following address:*

William James Hunt
Lattice and Microsoft C Users' Group
P.O. Box 271965
Concord, CA 94527

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

C CODE FOR THE PC

source code, of course

C Source Code

Bluestreak Plus Communications (two ports, programmer's interface, terminal emulation)	$400
Greenleaf Data Windows (windows, menus, data entry, interactive form design)	$315
Barcode Generator (specify Code 39 (alphanumeric), Interleaved 2 of 5 (numeric), or UPC)	$300
GraphiC 4.0 (high-resolution, DISSPLA-style scientific plots in color & hardcopy)	$275
Vitamin C (MacWindows)	$200
Greenleaf Communications Library (interrupt mode, modem control, XON-XOFF)	$160
Greenleaf Functions (296 useful C functions, all DOS services)	$160
Essential C Utility Library (400 useful C functions)	$160
Essential Communications Library (C functions for RS-232-based communication systems)	$160
PC/IP (CMU/MIT TCP/IP implementation for PCs)	$100
B-Tree Library & ISAM Driver (file system utilities by Softfocus)	$100
The Profiler (program execution profile tool)	$100
Entelekon C Function Library (screen, graphics, keyboard, string, printer, etc.)	$100
Entelekon Power Windows (menus, overlays, messages, alarms, file handling, etc.)	$100
QC88 C compiler (ASM output, small model, no longs, floats or bit fields, 80+ function library)	$90
CBTree (B+tree ISAM driver, multiple variable-length keys)	$80
ME (programmer's editor with C-like macro language by Magma Software)	$75
Wendin PCNX Operating System Shell	$75
Wendin PCVMS Operating System Shell	$75
Wendin Operating System Construction Kit	$75
EZ_ASM (assembly language macros bridging C and MASM)	$60
Multi-User BBS (chat, mail, menus, sysop displays; uses Galacticomm modem card)	$50
Heap Expander (dynamic memory manager for expanded memory)	$50
Make (macros, all languages, built-in rules)	$50
Vector-to-Raster Conversion (stroke letters & Tektronix 4010 codes to bitmaps)	$50
Coder's Prolog (inference engine for use with C programs)	$45
PC/MPX (light-weight process manager; includes preemption and cooroutine packages)	$45
Biggerstaff's System Tools (multi-tasking window manager kit)	$40
TELE Kernel (Ken Berry's multi-tasking kernel)	$30
TELE Windows (Ken Berry's window package)	$30
Clisp (Lisp interpreter with extensive internals documentation)	$30
Translate Rules to C (YACC-like function generator for rule-based systems)	$30
6-Pack of Editors (six public domain editors for use, study & hacking)	$30
ICON (string and list processing language, Version 6 and update)	$25
LEX (lexical analyzer generator)	$25
Bison & PREP (YACC workalike parser generator & attribute grammar preprocessor)	$25
C Compiler Torture Test (checks a C compiler against K & R)	$20
PKG (task-to-task protocol package)	$20
A68 (68000 cross-assembler)	$20
Small-C (C subset compiler for 8080 and 8088)	$20
tiny-c (C subsubset interpreter including the tiny-c shell)	$20
Xlisp 1.5a (Lisp interpreter including tiny-Prolog in Lisp)	$20
List-Pac (C functions for lists, stacks, and queues)	$20
XLT Macro Processor (general purpose text translator)	$20
C Tools (exception macros, wc, pp, roff, grep, printf, hash, declare, banner, Pascal-to-C)	$15

Data

DNA Sequences (GenBank 48.0 of 10,913 sequences with fast similarity search program)	$150
Protein Sequences (5,415 sequences, 1,302,966 residuals, with similarity search program)	$60
Webster's Second Dictionary (234,932 words)	$60
U. S. Cities (names & longitude/latitude of 32,000 U.S. cities and 6,000 state boundary points)	$35
The World Digitized (100,000 longitude/latitude of world country boundaries)	$30
KST Fonts (13,200 characters in 139 mixed fonts: specify TEX or bitmap format)	$30
NBS Hershey Fonts (1,377 stroke characters in 14 fonts)	$15
U. S. Map (15,701 points of state boundaries)	$15

The Austin Code Works
11100 Leafwood Lane
Austin, Texas USA 78750-3409

Voice: (512) 258-0785
BBS: (512) 258-8831
Email: FidoNet 1:382/12

Free surface shipping on prepaid orders

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

MasterCard/VISA

Number 30 on the Reader Service Card

The Proposed ANSI C Language Standard

by
Rex Jaeschke

© *Rex Jaeschke, 1987. All rights reserved.*

This is the tenth installment of Rex Jaeschke's ongoing report on the deliberations of the ANSI C Language Standards Committee. In this installment Rex begins to take a look at implementations of C in light of the evolving ANSI Standard.

const/volatile Correction

Jim Gimpel, author of C-terp and PC-Lint, called to question several examples in my column on the **const** and **volatile** type modifiers (*PJ*, Volume 5.4, July/August 1987). As Jim pointed out, two examples are incorrect. The last two lines on page 39 were

```
int const *cpi;
const int const *cpci;
```

They should have been

```
int * const cpi;
const int * const cpci;
```

The difference is that any **const** or **volatile** attributes that apply to the pointer must be written after the * in the declaration—something not allowed except for these new keywords. As written, the incorrect declarations **cpi** and **cpci** both declared pointers to **const int** objects. The compiler doesn't seem to object to seeing **const** twice before the * since one of them is redundant. I'm checking to see if this is the intent of the ANSI Standard.

The second error is at the top of page 40. The example

```
const volatile int const volatile *cvpcvi;
```

should be

```
const volatile int * const volatile cvpcvi;
```

Again, the problem is with the placement of the *. Thanks, Jim.

While we're on the subject, an interesting possibility arises with unions. For example,

```
union tag {
      const int ci;
      short s;
      volatile float vf;
};
```

Here, we have a union of a **const** object, a **volatile** object, and a non-**const**, non-**volatile** object. This is a valid construct although I'm hard pressed to find a use for it. The interesting part is that it allows you to modify the union via **s** and **vf** but not **ci**. Also, the **volatile** attribute applies only when the **vf** member is written into. The union itself does not have the **const** or **volatile** attributes—these would be specified when a union of type tag is declared. For example, a **const** union might have one or more **volatile** members.

A Look at Turbo C

Since the ads for Turbo C claim it is "ANSI C compatible," I thought I'd check it out. (Note that since there is not yet an ANSI C Standard, no one can claim ANSI conformance.) This is not intended to be a review of the whole product, just of its ANSI C features or lack thereof.

The interactive and command-line versions support the same basic options, so for purposes of this discussion I'll use the interactive version and its corresponding terminology. The Options command is the key to defining a compilation and linking environment that provides strict ANSI-type checking and enables the generation of portable code. Assuming that you want this kind of capability, let's look at the various options available and how you should use them.

The Compiler Options

Select the large memory model since it uses 32-bit pointers and 16-bit ints. If pointers and ints are the same size, then, depending on the compilation options you chose, some subtle bugs may escape detection.

You can define the signedness of char objects on a global basis or on an individual basis using the signed and unsigned type modifiers.

Do not merge duplicate strings.

Choose byte or word alignment as appropriate. (Note that the use of sizeof is paramount when trafficking in the sizes of structures and unions since recompilation with a different alignment may change the size of a structure or union.)

Make sure you test for stack overflow since it is not very easy to determine the actual amount of stack you need for a particular application. The amount is determined by the logic flow of the program, recursion, function call overhead, and other factors.

Turbo C allows identifiers to have 32 character significance. Use 31 since that is what ANSI requires for internal names. (Of course, you should use eight if porting to a K&R-style compiler.)

PRODUCED 2006 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Disable support for nested comments.

The "display messages and warnings" facility is excellent, allowing you to trap sloppy coding practices. Select all of them when you write new code, and then if you get no warnings during compilation, you should have some pretty portable code.

For maximum portability, make the link case-insensitive.

Enable Keywords only. This option defines the macro **__STDC__**, so you can conditionally compile based on whether or not you are in "ANSI mode" as defined by the Standard. The standard headers use this mechanism as follows:

```
#if __STDC__
#define _Cdecl
#else
#define _Cdecl cdecl
#endif
```

Note that **#if** is used rather than **#ifdef**. If ANSI mode is selected, **__STDC__** is defined and has a value of 1. If ANSI mode is not selected, **__STDC__** is not defined; however, if you use a macro in an **#if** expression and that macro is undefined, it defaults to the value 0. This is true only for the **#if** directive.

Trouble in Turbo City

As a matter of interest I decided to play with **__STDC__** and the four other predefined standard macros, and in the process I discovered a problem. I compiled the following program with ANSI keyword support, which causes **__STDC__** to be defined:

```
#include.h


main()
{
#ifdef __STDC__
    printf("__STDC__ is defined\n");
#else
    printf("__STDC__ is not defined\n");
#endif
    printf("__STDC__ is %d\n", __STDC__);
}
```

It compiled without error and gave the following output:

```
__STDC__ is not defined
__STDC__ is 1
```

These two statements contradict each other. If I compile without ANSI mode, **__STDC__** is not defined and the third **printf** (correctly) generates a compilation error to that effect.

I performed a similar test in ANSI mode for **__FILE__**, **__LINE__**, **__DATE__**, and **__TIME__** with the following results. (Note that **__LINE__** is a decimal integer and the other three macros are literal strings.)

```
__FILE__ is not defined
__FILE__ is E:\WRITING\PJOURNAL\N1.C
__LINE__ is not defined
__LINE__ is 17
__DATE__ is not defined
__DATE__ is Aug 25 1987
__TIME__ is not defined
__TIME__ is 12:38:13
```

It seems that even though these five predefined macros are actually defined and usable in ANSI mode, using them with the **#ifdef** directive indicates that they are not defined. This is an error.

A (possibly) related problem occurs with **cpp**, the standalone preprocessor. If you use it with the demo program **hello.c** as provided and you don't use the ANSI switch -A, line 8 of **stdio.h** produces an error. This line is **#if __STDC__**. It appears that **cpp** also has a problem dealing with predefined macros. The same problem occurs with other headers that include this line. While -A solves this problem, the output file contains garbage at the end of various lines even when using -P. You must use -P- to get the correct output. This is also an error.

Miscellaneous ANSI Features

The following ANSI features are supported:

- function prototypes (including dummy argument names)
- prototype-style function definitions
- signed keyword
- const and volatile type modifiers
- unary plus operator
- void functions
- concatenation of adjacent string literals
- the unsigned integral type suffix U (or u)
- the escape sequences \xhhh, \a, \v and \?
- the long double type
- long double and float constants
- generic pointers (void *)
- signed bit fields (permitted but not required by ANSI)
- #, ##, and defined preprocessor operators
- #include identifier
- #elif, #error, and #pragma directives
- %p in printf for displaying pointers

The following ANSI features are not supported:

- trigraphs (even though \? is supported)
- sizeof should not be permitted in preprocessor #if. It is.

The Runtime Library

All headers except **locale.h** (internationalization support) are provided. (**locale** is a recent addition and is still being finalized.) Although I didn't perform a thorough check, the headers looked to have the necessary stuff declared, etc., with a few exceptions. The **signal** function is not declared in **signal.h**—**ssignal** and **gsignal** are used instead. **float.h** is lacking a few macros, but these may have been added since Turbo C was written. (The technique in **limits.h** to detect whether chars are signed or unsigned is worth studying.)

Numerous headers contain macros, structure tags, and enumeration constants that do not begin with an underscore. Many of them should since they are used only internally: for example, **ctype.h** defines **IS_SP** (and a family of names **IS_***). Since ANSI C explicitly reserves the identifier namespace

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

starting with underscores for implementers, they should use it to avoid potential conflict with user identifiers.

My Conclusions

Page 197 of the *Turbo C User's Guide* states, "Turbo C not only supports the K&R definition, it also implements most of the ANSI extensions." My experience supports this claim so much so that I give Turbo C an A rating. To get an A+, Borland needs to fix a few problems (some of which I've mentioned above). The negative points I've mentioned are very minor and, clearly, Turbo C is well positioned for the final ANSI/ISO Standard. If portability and standard conformance are requirements, Turbo C is a prime contender. By the time you read this, though, Microsoft's Version 5.0 compiler will be shipping, and from what I've seen of it, it will provide some stiff competition.

Microsoft C V5 and QuickC

I have a Beta-test version of Microsoft's Version 5.0 and QuickC, and under their test program rules, I receive a new set of disks (and often manuals) every two weeks or so. While the products seem to be relatively stable and set regarding functionality, I intend to delay an ANSI-feature review until final versions are shipped. The primary reason for this is that being on the ANSI C Committee I am well aware of the dark corners of C and the (often subtle) ways in which the language is impacted by the proposed Standard. I want to get their final releases before I really "put them through their paces." To do so any earlier would not be fair to Microsoft or my readers. For the interim, suffice it to say that this version will compete favorably with Turbo C.

New Releases Expected Soon

I expect to have copies of Instant-C Version 3.0, DeSmet C Version 3.0, and PC-Lint Version 2.10 in time for the next issue. All of these provide varying levels of support for the proposed standard.

Future ANSI C Meetings

The schedule for future ANSI meetings is: December 7–11, Austin, Texas (note that this replaces the meeting previously scheduled in January for Austin); April 18--22, 1988, Nashua, New Hampshire. ♦

Rex Jaeschke is an independent computer consultant, writer and lecturer. He is also a member of the X3J11 C Standard Committee, observer on the ISO C Standard Committee, Founding Editor of The C Journal, *and the C language Editor of* DEC PROFESSIONAL.

If you would like to comment on or question any aspect of the Proposed ANSI C Standard, write to

Rex Jaeschke
2051 Swans Neck Way
Reston, VA 22091

Rex will address your concerns in future columns as space permits.

"If you need source code, make sure your wallet is wide open or get VITAMIN C."
- Computer Language, June, 1987

Vitamin C

It's good for your system!

- Professional C function library
- 30 day money back guarantee
- Multiple bullet proof windows
- Easy full screen data entry
- Unlimited data validation
- Context sensitive help manager
- Menus like Lotus and Mac
- Programmable keyboard handler
- Text editor routines
- No royalties or runtime fees
- Library source included FREE
- Free technical support
- Free BBS at (214)418-0059
- Supports all major compilers including Microsoft 5.0
- VCScreen code generator too!
- UNIX version avaialable

Fast, flexible, versatile, reliable. Vitamin C delivers the vital combination that software professionals demand. Highly efficient, professionally crafted C code provides lightning fast displays required by today's window intensive programs.

High level functions provide maximum productivity and require little supporting code. Plus, Vitamin C's versatile, open ended design is full of hooks so you can intercept and plug-in special handlers to customize or add features to most routines.

VCScreen, our screen painter / code generator speeds your development even more! Simply draw your input forms using our interactive design editor and generate perfect C source code ready to compile and link with the Vitamin C library.

Vitamin C $225
Includes all source code FREE! For IBM PC, XT, AT, PS/2 and true compatibles. Specify compiler when ordering.

VCScreen $99.95
For IBM PC, XT, AT, PS/2 and true compatibles. Requires Vitamin C.

ORDER NOW!

We ship UPS. Please include $3 for ground, $6 for 2-day air, $20 for overnight, or $30 if outside the U.S. Texas residents add 7 1/4 % sales tax. All funds MUST be in U.S. dollars drawn on a U.S. bank. Visa & MasterCard accepted.

(214)416-6447

creative PROGRAMMING

Box 112097 • Carrollton, Texas 75011

Windows • Data Entry • Menus • Help • Text Editing
Plus... All Source Code FREE!

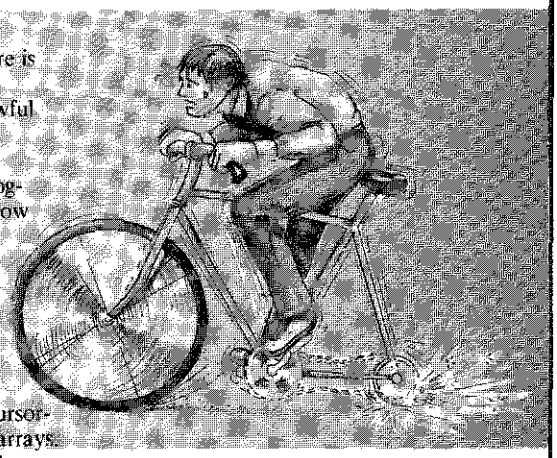
Contact this Advertiser directly

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

If You Have Turbo C You Have Half Your C-Programming Vehicle

Turbo C is a great compiler but there is one vital cog missing—debugging. Without it, you have to spend an awful lot of energy to go a short distance.

Gimpel Software's C-terp, long recognized as the leading C interpreter, now fully supports Turbo C with complete compatibility guaranteed.



The missing wheel that will turn your half-cycle into a bicycle

Interactive Debugger—Our debugging facilities include split screen (code in upper portion, dialog in lower), breakpoints (sticky, temporary, line/function, cursor-directed), display of structures and arrays, execution of any expression (even those involving macros), function traceback with arguments, watch expressions and watch conditions (watchpoints). Our watch expressions can be structs or arrays. We catch out-of-bounds pointers!

No Toy—Full K&R with ANSI enhancements. Multiple-module with a built-in automatic make. It has virtual memory option (with optional direct use of extended memory) and a shared symbol option for those big programs. It supports graphics, dual displays and the EGA 43-line mode.

Links to external libraries—(both code and data, automatically) which can call back to interpreted functions. Function pointers are compiler compatible.

100% Turbo-C compatible.—Same header (.h) files, data alignment, bit field orderings and preprocessor variables as your compiler. We link in your compiler's library.

Our reconfigurable editor—is multifile and comes with a configuration script to mimic Turbo's editor.

C-terp

Order C-terp today!

Call (215) 584-4261

Introductory Price for Turbo C-terp:
$139.00

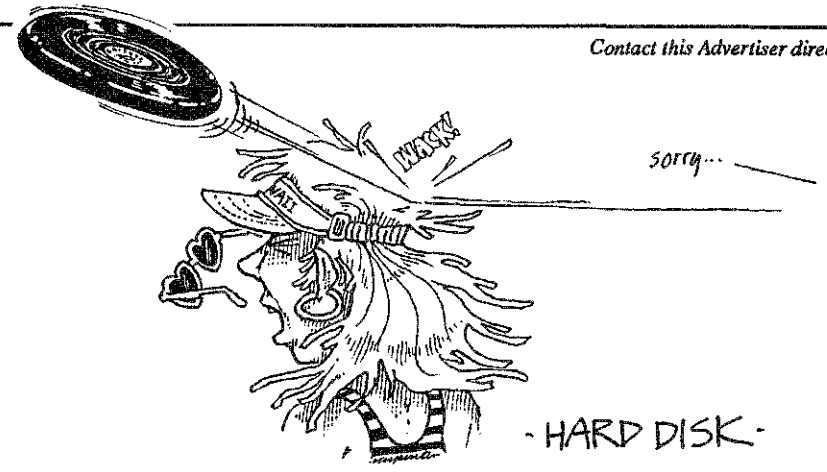
VISA, MC, COD—30 day money back guarantee

C-terp Version 3.0 is also available for the following compilers:
Microsoft, Lattice, Aztec, C86, and Mark Williams ($298) and Xenix ($498).

GIMPEL SOFTWARE
3207 Hogarth Lane
Collegeville, PA 19426

C-terp is a trademark of Gimpel Software, and Turbo C of Borland International.

Contact this Advertiser directly



C PROGRAMMER!

10 Important Reasons to Use db_VISTA™ for File Management

1. **db_VISTA is written in C.**
2. **It's fast,** combining B-tree indexing with "network" or direct "set" relationships between records.
3. **It's flexible.** Use **db_VISTA** as a file manager or a complex database; single-user PC to multi-user VAX with millions of records. Upgrade easily with full compatibility!
4. **It's portable.** MS-DOS, UNIX, VMS . . . see list below.
5. **It uses space efficiently.** Non-RAM resident: only operative functions are in your run-time program.

And there's more . . .

6. **Royalty-free run-time.**
7. **Complete source code available.**
8. **SQL-based db_QUERY!** Add fast, C-linkable db_QUERY for ad-hoc queries and reports. Build an end-user interface; provide a relational view of the database.
9. **FREE tech support hotline!** 60 days free; for product or application development help. Extended support, consulting, training classes — all available.
10. **30 Day Money-Back Guarantee.** Full refund if not completely satisfied.

That's just the beginning!
Want more details? Call today!

Order Now. It's easy — simply call toll-free. We'll answer your technical questions and get you started . . . fast delivery.

Call Toll-Free Today!
1 (800) db-RAIMA
(that's 1-800-327-2462)
— OR Call 1-206-828-4636

db_VISTA™ Royalty Free Prices

	Object	w/Source
Single-User	$ 195	$ 495
Multi-User	$ 495	$ 990
VAX Multi-User	$ 990	$1,980

db_QUERY prices are the same as above

MasterCard VISA

- **Operating systems:** MS-DOS, UNIX, XENIX, ULTRIX, VMS; more . . .
- **Compilers:** Microsoft, Lattice, IBM, Computer Innovations, Aztec, Turbo C, UNIX, XENIX; more to come . . .
  also works with most C libraries!

RAIMA™ CORPORATION

3055 - 112th N.E., Bellevue, WA 98004 USA
(206) 828-4636 Telex: 6503018237 MCI UW

PJ 117

Contact this Advertiser directly

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Windowing and Graphics

GEM Developer Toolkit, Release 2
Digital Research
Technical Support Department
Box DRI
Monterey, CA 93942
800/443-4200
$500

Two major competitors for graphics windowing environments under MS-DOS are GEM (Graphics Environment Manager) from Digital Research and Windows from Microsoft. Ventura Publisher, Xerox's premier page composition program, is based on GEM while Pagemaker for the PC from Aldus Corporation uses Microsoft Windows. Benchmarks comparing current versions of these two environments have typically shown GEM to be much faster for graphics. GEM started out many years back as GSX under CP/M-80, a graphics extension to the standard 8-bit operating system. GSX then migrated to the 8086 as GSX-86 for CP/M-86. These earlier products evolved into GEM for the 8086 chip family (MS-DOS and Concurrent DOS) and the GEM-68k for the 680x0 (Atari ST).

The GEM graphics environment supports icons, pull-down menus, dialog boxes, windows, pointing devices, and fonts on a variety of hardware devices. GEM software works independently of operating systems and languages and is compatible with PC-DOS, MS-DOS, and Concurrent DOS. The GEM VDI (Virtual Device Interface) supports two coordinate modes: normalized device coordinates (NDC—must map coordinate space onto physical coordinates) and raster coordinates (bit maps). This raster mode enables fast and efficient windows implementation.

The GEM Developer's Kit provides programming tools and documentation to create GEM applications. The toolkit consists of two PC style programming and reference manuals (600+ pages), GEM 2.2 software and sample applications, and development and utility software. The 8086 version has support for Lattice C Versions 2.x and 3.0 and Microsoft's Assembler (MASM). A GEM version of SID-86 (DRI's symbolic debugger) is included along with utilities to translate map files from either LINK or PLINK (Phoenix). The manuals are mostly processor and operating system independent. Brief appendices cover processor-specific data for the 8086 and 68000.

A GEM 86 Device Driver Supplement is available at additional cost to facilitate writing hardware device drivers. A GEM Developer Kit for the 68k is available from Atari. Technical support is provided via a restricted CompuServe forum and on-line phone assistance. Redistribution licenses for using GEM system software and device drivers must be negotiated.

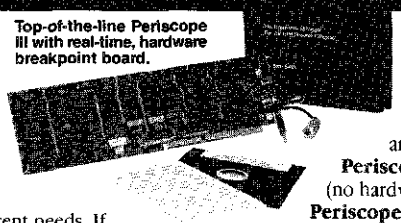
OS/2 Graphics Development Toolkit
Graphic Software Systems
9590 SW Gemini Drive
Beaverton, OR 97005
503/641-2200
OS/2 $995, DOS $495

Graphics Software Systems (GSS) is another significant player in the microcomputer graphics arena. Founded in 1981, GSS developed GSX with Digital Research. GSS has been the developer of the IBM Graphics Development Toolkits under DOS, OS/2, and Xenix. GSS is now shipping a new toolkit for OS/2 and an updated toolkit for MS-DOS. These toolkits provide a device independent interface to graphics devices without the performance overhead of what has been called the "WIMP" (Windows, Icons, Mouse, and Pull-down menus) interface. The toolkits are based on the Computer Graphics Virtual Device Interface Baseline Document (CGI) under development by both ANSI and the International Standards Organization (ISO). These evolving standards provide source code portability and device independence to minicomputer and mainframe environments. The GSS*CGI software operates with a library of device drivers and a choice of interfaces to high level languages.

The advance release of the GSS OS/2 Graphics Development Toolkit (GDT) is source code compatible with the DOS Graphics Development Toolkit. The OS/2 toolkit can be used to port existing graphics packages to the protected mode environment, as well as to develop new OS/2 graphics applications. The current OS/2 GDT includes language bindings to support only Microsoft C and Macro Assembler (MASM). The final version will support OS/2 implementations of C, BASIC, FORTRAN, Pascal, and assembler from IBM and third party vendors. This release includes only limited device drivers for VGA, EGA, and CGA graphics adapters and the Microsoft mouse. Output devices supported include IBM's Proprinter, Graphics Printer, Color Graphics Printer, and Quietwriter III printer and plotters from IBM and Hewlett-Packard. Additional drivers are under development including the IBM PS/2 Mouse, the HP Laserjet+ printer, and laser and dot matrix printers from Epson and other vendors. Discounts are available to registered owners of the GSS GDT for DOS. The final

PERISCOPE™ POWER

...Keeps you going full steam ahead when other debuggers let you down. With four models to pick from, you'll find a Periscope that has just the power you need.



. . . Start with the model that fits your current needs. If you need more horsepower, upgrade for the difference in price plus $10! And don't worry about having a lot more to learn . . . Even when you move to the most powerful model, Periscope III, an extra dozen commands are all that's involved.

Periscope's software is solid, comprehensive, and flexible. It helps you debug just about any kind of program you can write . . . thoroughly and efficiently. Periscope's hardware adds the power to solve the really tough debugging problems.

Periscope requires an IBM PC, XT, AT, or close compatible (Periscope III requires hardware as well as software compatibility); DOS 2.0 or later; 64K available memory; one disk drive; an 80-column monitor.

Periscope I includes a half-length board with 56K of write-protected RAM; break-out switch; software and manual for $345.

Periscope II includes break-out switch; software and manual for $175.

Periscope II-X includes software and manual (no hardware) for $145.

Periscope III includes a full-length board with 64K of write-protected RAM, hardware breakpoints and real-time trace buffer; break-out switch; software and manual. Periscope III for machines running up to 8 MHz is $995; for machines running up to 10 MHz, $1095.

Call Toll-Free for free information or to order your Periscope today!

MAJOR CREDIT CARDS ACCEPTED. **800-722-7006**

The PERISCOPE Company, Inc.
14 BONNIE LANE
ATLANTA, GA 30328
404/256-3860

Number 6 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

version is to be priced at $695.

The new release of the DOS GDT includes additional device drivers for PS/2 graphics devices. The DOS toolkit supports a larger range of devices including Hercules monochrome graphics, HP Laserjet+ and Thinkjet printers, Toshiba 351 and Epson 80 and 100 printers, Diablo C-150 and Tektronix 4695 color printers, Mouse Systems mouse, Summagraphics tablet, and Ricoh IS30 Scanner. Language bindings are provided for Lattice and Microsoft C, Microsoft and Ryan-McFarland FORTRAN, Microsoft Pascal, QuickBASIC, and Assembler. Low cost updates are available.

How do these versions differ from toolkits to be sold by IBM? Only in minor aspects—for example, circles are drawn clockwise instead of counter-clockwise. Are you left or right handed? Redistribution licenses for either toolkit must be negotiated.

Curses Window Development Package
Aspen Scientific
P.O. Box 72
Wheat Ridge, CO 80034-0072
303/423-8088
binary $119, source code $289

Aspen Scientific has released Version 3 of Curses Window Development Package for MS/PC-DOS 2.0 and later. The Unix programming environment provides the developer with many tools for improving software quality and programmer productivity, and Curses is one of these software tools. Curses provides a text only windowing environment available under Unix. The Curses library routines allow the quick development of user screen and keyboard interfaces that are portable to most character oriented terminals. A terminal interface is defined in a database called Terminfo. If a terminal is defined in the database, Curses can be used with it. A programmer can define multiple windows and respond to function and special keystrokes without knowing the details of the hardware environment.

The Aspen Curses implementation consists of a fairly complete library of 130+ functions and macros for handling characters and windows on the screen and keyboard events. A few Unix functions are not provided because they either serve no purpose on the PC or are not portable. The package also provides some extensions to Unix Curses library functions specific to the IBM PC and compatibles. These Curses routines should also support less compatible MS-DOS display hardware via BIOS routines or as an ANSI terminal. The Aspen Curses library supports IBM monochrome, CGA, and EGA color monitors along with a DOS ANSI terminal driver. The terminal type is changed easily using an environment variable. This toolkit provides a simple way to port Unix programs that use Curses to the PC world. It provides software developed for the PC upward compatibility with the Unix market and can also provide portability among a variety of PC hardware. The Curses routines have been written in assembly language for speed. The routines are provided as object libraries for a variety of C compilers including Microsoft, Lattice, Turbo C, and Aztec. Source code is also available for an additional charge.

Aspen also includes the FAST Forms Developer Tool Kit. FAST is an editor and C language interface for creating and using data input forms. FAST comes with a library of C routines in source code. The tool kit relies heavily on Curses for device independent video and keyboard support. The reasonable license agreement allows incorporating these routines into unlimited user programs for distribution.

continued

TASKVIEW
ROCK SOLID MULTITASKING!

Packed with the power you need, TASKVIEW takes you beyond the limits of DOS.

- Communicate while you edit
- Compile while you print
- Load up to 5 megabytes of programs
- Manage resident utilities
- Time-slice multiple jobs
- Cut & paste between programs

TASKVIEW lets you load up to 10 of your favorite applications and switch between them at a keystroke. They can even continue to run while you work on something else! TASKVIEW is the BEST multitasker you can buy. Just ask our customers.

"Thanks for a wonderful product!"

"TASKVIEW is the only multitasker I own which runs efficiently & correctly both on my Zenith Z-151 . . . and on my Orchid 286e . . . I also have DoubleDOS, DESQview, TopView, Windows, and Concurrent PC-DOS".

"Fantastic! What DoubleDOS, MS-Windows, and all the other 'stuff' should have been!"

"My BBS is now up 24 hours, 7 days a week thanks to TASKVIEW."

"Thanks! I needed that!"

TASKVIEW requires an IBM PC, XT, AT or Jr compatible, and PC or MS DOS 2.x to 3.x. To get your copy, call toll free:
(800) 367-0651
or send $79.95 + $3.00 S&H ($8.00 Intl.) to:

Sunny Hill Software
PO Box 55278, Seattle, WA 98155-5278
For more information call (206) 367-0650
DoubleDOS trademark Softlogic Solutions, Concurrent PC-DOS reg. trademark Digital Research Corp., MS Windows trademark Microsoft Corp., DESQview trademark Quarterdeck Office Systems, Inc. TopView trademark IBM Corp.

Number 85 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

STOPWATCH

MEASURE EXECUTION TIME TO THE MICRO-SECOND

■ ***STOPWATCH™ . . . It Measures Program Time with Microsecond Resolution.*** The new STOPWATCH Program measures total and component times of programs executed on IBM PCs, XTs, ATs, and compatible units to within ± 5 microseconds.

■ ***STOPWATCH . . . It's Easy to Use.*** With STOPWATCH there's no need to modify your program for testing. Simply key ⟨ctl⟩-M instead of ⟨enter⟩ anytime you want to measure execution time to the next display output. (⟨ctl⟩-M and ⟨enter⟩ are equivalent *until* you install STOPWATCH.)

■ ***STOPWATCH . . . Use It To Time New Programs.*** Use STOPWATCH with DEBUG, SYMDEB or CODEVIEW to measure execution time from "GO" to Breakpoint. And, STOPWATCH can be used to time Lotus 1-2-3 Macros, dBASE III Procedures, BASIC Programs, or any object program that you have developed.

■ ***STOPWATCH . . . Use It To Evaluate Your Present System.*** Add STOPWATCH to your present system and measure component times of a program to quickly determine whether a new processor, or faster disk, is the key to improving response time for your application.

■ ***STOPWATCH . . . Unparalleled Accuracy At An Affordable $59.95.*** Included are demo looping programs to demonstrate STOPWATCH's timing ability, the STOPWATCH Program, and a user's manual with easy-to-follow instructions to get you started quickly. Now . . . Microsecond resolution timing — only $59.95! Put your program to the test today with STOPWATCH!

Now Microsecond Resolution Timing Only $59.95 + 3.50 S&H + 6% NJ Tax. MC/VISA accepted

US 800-345-0167
NJ 201-228-7623

CRTS
CUSTOM REAL-TIME SOFTWARE

P O Box 1106, West Caldwell, NJ 07007

IBM is a registered trademark of International Business Machines Corp.
CodeView is a trademark of Microsoft Corp
Lotus and 1-2-3 are registered trademarks of Lotus Development Corp.
dBase III is a trademark of Ashton-Tate
*STOPWATCH is a Trademark of Custom Real-Time Software Inc

Number **124** on the Reader Service Card

PC TECH 34010 Graphics Boards
PC Tech, Inc.
904 North 6th Street
P.O. Box 459
Lake City, MN 55041
612/345-4555
Mono $595 (w/monitor $1295)
Color $1199 - $1499

This competent firm has been designing and manufacturing advanced PC hardware at reasonable prices for several years. I have recently seen their Graphics Interface Boards based on the Texas Instruments TMS34010 32-bit Graphics Processor. They currently manufacture two boards for the PC. The monochrome board comes with a full page vertical monitor and supports resolution up to 800 x 1024 pixels. The board is capable of 1024 x 1024 resolution. It emulates the IBM CGA and Monochrome video boards and the MDS GENIUS system. The board contains 256K of graphics memory and 256K of 34010 memory for video buffers, 34010 program storage, and fonts and windowing applications. Up to 32K bytes of EPROM are available. The board runs Microsoft Windows, Halo, GEM, etc.

The color board supports resolutions to 1024 by 800 pixels with 256 simultaneous colors from a palette of 256K using an analog monitor. This board comes with up to 1 megabyte of graphics memory and up to 4 megabytes of 34010 memory for video buffers, program storage, and fonts and windowing applications. Up to 512K of EPROM is supported. The color board emulates the IBM CGA initially although other emulations are under development. This board is designed to bring workstation quality graphics and performance to the PC. With the speed and programmability of the TMS34010, virtually any video board may be emulated, but even higher performance may be achieved using a high level driver such as CGI, GKS, or DGIS. Supplied with a Microsoft Windows driver, both boards perform with true workstation speed and resolution. Either board occupies one slot in an IBM XT/AT or compatible and comes with a year of free software updates. This company also makes compact cards with 2 to 16 megabytes of expanded/extended memory for the IBM XT/AT.

Zsoft Technical Reference Manual
Zsoft Corporation
1950 Spectrum Circle, Suite A-495
Marietta, GA 30067
404/980-1950
free

If you are working in a graphics environment, you might want to be able to import and export files with other programs. Zsoft is the publisher of PC Paintbrush, one of the most popular (MacPaint like) paint programs available under MS-DOS. This is the paint software distributed with the the Microsoft and Logitech mouses. Programs such as Ventura Publisher support importing PC

'Faster and better in no time'

Create better, faster, higher quality and easier to read progams in a fraction of the time. Let your system do the work for you. With 23 powerful, state of the art tools for the IBM PC and compatibles, both beginners and experts will find programming a breeze.

- Powerful, common user interace
- Interactive and batch execution
- Online documentation
- Self-explanatory error messages

C **Programmer's Toolbox**
Volumes I & II

- Monitor program/system execution
- Beautify program listings
- Determine program flow/critical path
- Trace/verify variable usage
- And more....

At $79.95 per volume or $130 for both with a **30 day trial**, the Toolbox is simply the **best value** today. In addition, the documentation is packed with examples and tips on creating better programs.

Why waste time, **call or write us today.**

™

MMC AD Systems
Box 360845 Milpitas, California 95035
(408) 263-0781

Number **94** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Paintbrush picture files directly. Zsoft publishes a short reference manual (18 pages) describing their picture and font file formats. The manual also describes how to invoke Frieze (their memory- resident graphics device driver) from within other programs. Brief example C subroutines are provided for reading and writing picture files. ZSoft also distributes a quarterly newsletter describing their products and supported hardware. Both of these publications are free for the asking.

Compilers and Assemblers

DeSmet C Version 3.0
C Ware
P.O. Box 428
Paso Robles, CA 93447
805/239-4620
$109

C Ware has released Version 3.0 of the DeSmet C Compiler. DeSmet C was already one of the fastest C compilers available under DOS. The latest version compiles much faster than previous versions, cutting 15 to 40% off compile times. The assembler pass of the compiler is normally bypassed, and object code is directly output. In addition, a Large Case option allows programs to use the large memory model of the 8086. Version 3.0 also includes an expanded stdio library. The basic compiler package ($109) includes the C Compiler, Assembler, and Linker along with several utility programs. The compiler supports inline assembly language. The utilities include a fast customizable full screen programmer's editor (SEE) that supports two windows and unlimited file sizes and a C listing and cross reference utility. Compiler options include the large memory model ($50) and a solid source code debugger ($50).

MASM V5.0
Microsoft Corporation
16011 NE 36th Way
Box 97017
Redmond, WA 98073-9717
206/882-8080
$150

Microsoft has released Version 5.0 of their Macro Assembler and utilities. This version is supposed to be 25 to 40% faster than MASM 4.0 for assembling and even faster for linking. This release also supports the 80386/80387 microprocessors and includes Codeview, their source code level debugger. The manuals (1000 pages+) have finally shed their binders and come as three perfect bound paperbacks (like Borland's) and a smaller spiral bound reference guide. The reference guide includes a nice 80xxx instruction set summary with instruction mnemonics and timings for the various chips from 8088 to 80387. The documentation includes a special guide (140 pages) for mixed-language programming with Microsoft products.

continued

SemWare Announces QEdit Advanced the "quick editor"

QEdit is one of the smallest, fastest, multi-file editors available for IBM compatible computers and has many features to enhance your productivity.

- Lightning fast, especially in file loading and saving, screen display, cursor movement, and global searching and replacing.
- Edit multiple files, as many and as large as available memory allows. QEdit can use all available memory, up to 640KB. Ideal for multi-file languages and all those .PRG files.
- Windowing allows up to 7 files or 7 views of the same file to be seen at once.
- You do not have to "learn" another editor with QEdit. QEdit's commands can be configured to any keys you like.
- Screen colors, tab settings, default file type and many other options can be configured the way YOU like it.
- Up to 99 named scratch buffers may be created and edited. Commands are available for easy cut and paste operations.
- Invoke a DOS "shell" or execute DOS commands directly from QEdit.
- Both line and stream block marking are supported.
- Keyboard macros allow automation of commonly used command sequences or text entry.
- Commands to shift an entire block left or right, to adjust indentation levels when using structured languages.
- Garbage stack provides restoration of the last "n" (user-settable) deletions.

Introductory Price
49.95
Mastercharge & Visa Accepted
Add $3.00 Shipping
Georgia Residents please include 5% sales tax

QEdit Advanced requires an IBM PC/XT/AT or Compatible, IBM PS/2 30 - 80, 128KB of available memory. Please specify disk format required.

To Order: Heritage Papers, Danielsville, GA 30633
Phone Orders: (404) 789-3226

Number **148** on the Reader Service Card

PORTABLE POWER WITH MasterFORTH

Whether you program on the **Macintosh**, the **IBM PC**, an **Apple II** series, a **CP/M** system, or the **Commodore 64**, your program will run unchanged on all the rest.

If you write for yourself, MasterFORTH will protect your investment. If you write for others, it will expand your marketplace.

Forth is interactive – you have immediate feedback as you program, every step of the way. Forth is fast, too, and you can use its built-in assembler to make it even faster. MasterFORTH's relocatable utilities and headerless code let you pack a lot more program into your memory. The resident debugger lets you decompile, breakpoint and trace your way through most programming problems. A string package, file interface and full screen editor are all standard features. And the optional target compiler lets you optimize your application for virtually any programming environment.

The package exactly matches *Mastering Forth* (Brady, 1984) and meets all provisions of the Forth-83 Standard.

MasterFORTH standard package	$125
(Commodore 64 with graphics)	$100
Extensions	
Floating Point	$60
Graphics (selected systems)	$60
Module relocator (with utility sources)	$60
TAGS (Target Applic. Generation System) – MasterFORTH, target compiler and relocator	$495
Publications & Application Models	
Printed source listings (each)	$35
Forth-83 International Standard	$15
Model Library, Volumes 1-3 (each)	$40

(213) 821-4340

MICROMOTION
8726 S. Sepulveda Bl., #A171
Los Angeles, CA 90045

Number **13** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

TRUE
MULTITASKING
With
MultiDos Plus
"multitasking for the IBM-PC."

Ideal for developing applications in process control, data acquisition, communications, and other areas. Check these features which make **MultiDos Plus** an unbeatable value.

- Run up to 32 programs concurrently.
- Your software continues to run under DOS. No need to learn a new operating system.
- Use the compilers you already have. Supports software written in most languages.
- Operator commands to load/run programs, change priority, check program status, abort/suspend/resume programs.
- Programmatic interface via INT 15H for the following.
  - * Intertask message communication. Send/receive/check message present on 64 message queues.
  - * Task control by means of semaphores. Get/release/check semaphores.
  - * Change priority-256 priority levels.
  - * Suspend task for specified interval.
  - * Spawn and terminate external and internal tasks.
  - * Disable/enable multitasking.
  - * and more!
- Independent foreground/background displays.
- Access to DOS while applications are running.

Hardware/Software Requirements

IBM PC/XT/AT or true clone. Enough memory to hold **MultiDos Plus** (48 KB) and all your application programs. Also may need 4 or 16 KB memory for "hidden screens" for each active task. MS-DOS (or PC-DOS) 2.0 or later operating system.

only: **$24.95** *OR*
$99.95
with source code

Outside USA add $5.00 shipping and handling.
Visa and Mastercard orders only call toll-free: 1-800-872-4566, ext. 350., or send check or money order to:

NANOSOFT
13 Westfield Rd, Natick, MA 01760
MA orders add 5% sales tax.

Number 109 on the Reader Service Card

Microsoft has finally simplified some of the directives for specifying segments and memory models. This version can use all of available memory, supporting up to 8 times as many symbols as MASM 4.0. Codeview Version 2.0 now emulates the Windows 2.0 overlapping window interface and supports Microsoft MASM 5.0, C 4.0 and 5.0, FORTRAN 4.0, and QuickBASIC 4.0. Minimum system requirements are DOS 2.0 and 256K of memory. It still runs on a lowly IBM PC. Upgrades from MASM 4.0 are $40, and from earlier versions are $75.

T-Debug PLUS
Turbo Power
3109 Scotts Valley Drive, Suite 122
Scotts Valley, CA 95066
800/538-8157 x830 (US)
800/672-3470 x830 (CA)
$60 (Upgrades $20)

T-DebugPLUS, the leading symbolic runtime debugger for Turbo Pascal, has been upgraded to Version 2.0. Watch variables, conditional breakpoints, backtracking, graphics support, and many more new features will help Turbo Pascal programmers find and fix bugs faster than ever before. New features allow opening a watch window on variables or a memory area. T-DebugPLUS can be used to debug programs that use CGA, EGA, and Hercules graphics modes even on a single monitor (dual monitor operation is also supported). The program can generate MAP files that are fully compatible with external debuggers such as Periscope, Atron, Symdeb, and others. The program comes with the complete T-DebugPLUS source code (written in Turbo Pascal), which may be studied or customized. All TurboPower software products are backed by a 30-day money-back guarantee.

Turbo Development System
MicroHelp, Inc.
2220 Carlyle Drive
Marietta, GA 30062
404/973-9272
$39 plus $3 shipping

The Turbo Development System (TDS) is a tool designed to work together with Borland's Turbo Pascal 3.xx compiler. TDS is a complete source code writing environment that has a very high level of compatibility with Turbo Pascal. The package consists of a Turbo compatible editor environment and a fast one-pass syntax checker for Turbo Pascal. TDS is designed as a tool for programmers writing larger applications in Turbo Pascal. TDS is for the user who feels frustrated by Turbo Pascal's insistence on recompiling entire programs again, even for the most minor mistakes. Error correction and source code modification are handled within the TDS editor environment. When a syntax error is detected, the editor automatically places the cursor at the location of the error. The programmer can then choose to correct the error (resuming syntax checking from that point), continue the checking without making a correction, or stop the correction process. TDS also includes useful SHELL, DELETE, RENAME, and COPY commands. All MicroHelp products are offered with a 30-day money-back performance guarantee.

Turbo Pascal Source Code Generator
C.C. Software
1907 Alvarado Avenue
Walnut Creek, CA 94596
415/939-8153
MS-DOS $67.50, CP/M-80 $45.00

Have you ever wanted to know what makes Borland's compiler technology tick? Is there some "minor" Turbo Pascal bug you've always wanted to eliminate? This small software company sells an honest to goodness Turbo Pascal disassembler that generates fully commented source code ready for change and re-assembly. You must have a copy of Turbo Pascal 3.01A to generate the source code from. I have seen the output of the CP/M-80 product, and it's about as well commented assembly language as you'll find anywhere. A fast assembler is included that re-assembles the 16,000 lines of code to an MS-DOS COM file in 20 seconds on an IBM AT.

80386 Hardware/Software

mW1167 Numeric Coprocessor Card
Microway
P.O. Box 79
Kingston, MA 02364
617/746-7341
Hardware $1495

An evolution of 80386 motherboards focused on memory and number crunching capabilities has been quietly taking place. The Intel 80836 chip was designed to support three numeric coprocessor chips: (1) the 80287, (2) the 80387, and (3) the Weitek numeric chip set. The first cycle of 386 motherboards (Compaq, ALR, etc.) were designed for expediencey to accept the 80287 chip set, but in the normal configuration, the 80287 is clocked the same as the I/O bus at 8 MHz. For number crunching, an inexpensive 8 or 10 MHz turbo PC clone with an 8087 (and a Nec V20/30) can be just as fast at a much lower cost. The second generation of 386 motherboards are designed to accept the 68 pin 80387 chip, which is clocked at the same speed as the CPU (16 MHz typical). This combination delivers at least twice the numeric speed of the first generation boards.

Now the next generation motherboards that support the 121 pin Weitek chip set for number crunching are beginning to arrive. This coprocessor can improve number crunching by 300 to 500% over the 80387.

Continued on page 75

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

DEVELOPMENT TOOLS FOR C

- A COLLECTION OF "TOOL BOXES" FOR THE DEVELOPMENT OF POWERFUL C PROGRAMS
- DESIGNED TO BE USED BY BOTH NEW AND EXPERIENCED C PROGRAMMERS
- EACH TOOL BOX IS FULLY DOCUMENTED -- WITH EXAMPLES

SYSTEM TOOL BOX
- ACCESS THE POWER OF YOUR PC
- KEYBOARD & SCREEN MANAGEMENT
- GRAPHICS & MUCH MORE
- 184 PAGE USER'S GUIDE

LIB + SRC $35

UTILITY TOOL BOX
- GENERAL PURPOSE FILTERS
- UTILITY PROGRAMS
- GENERAL PURPOSE FUNCTIONS
- 96 PAGE USER'S GUIDE

LIB + SRC $25

WINDOW TOOL BOX
- COMPLETE WINDOW CONTROL
- BUILD WINDOW DATABASES
- FULL ERROR REPORTING
- 84 PAGE USER'S GUIDE

LIB $45 -- SRC $125

MENU TOOL BOX
- VERTICAL/HORIZONTAL MENUS
- CREATE MENU DATABASES
- FULL ERROR REPORTING
- 78 PAGE USER'S GUIDE

LIB $35 -- SRC $75

- HARDWARE -- IBM PC, TI PRO OR TANDY 1000
- COMPILER -- MICROSOFT™ C (3.00 & LATER)

- CALL OR WRITE FOR FREE INFO & DEMO PROGRAM
- TECHNICAL SUPPORT IS UNLIMITED & FREE
- NO ROYALTIES

FLEMING SOFTWARE
5741 Wagonwheel Trail
Castle Rock, CO 80104
(303) 688 - 3584

Number 118 on the Reader Service Card

Coming Soon

The only code-intensive publication on C for MS-DOS® systems.

If you're serious about C and tired of wading through pages of ads, chatter or programs in other languages, turn to The C Gazette.

- *Code-intensive: pages of clear, fully-commented and portable C. Extensive textual support for the code.*
- *In-depth coverage of algorithms: detailed explanations presented from different perspectives by several authors.*
- *No-nonsense advertising policy: no products accepted for advertisement without prior examination by The C Gazette. No ads for vaporware or non-performing products.*

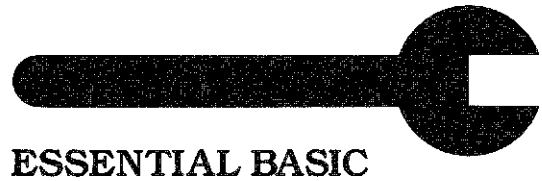
1 year (4 issues) $21. 2 years (8 issues) $38.

The C Gazette

1341 Ocean Ave. #257 Santa Monica, CA 90401 (213) 473-7414

Number 144 on the Reader Service Card

EXIM TOOLKIT:



ESSENTIAL BASIC PROGRAMMING TOOLS

A **must** for all BASIC programmers who use Microsoft's QuickBASIC or IBM's PC BASIC compilers, our feature-rich, easy-to-use **EXIM ToolKit** provides dozens of useful routines unavailable in BASIC or any other single software library sold today!

Here's just some of what the **EXIM ToolKit** can do:

With our **Screen Management** module, you get features like four-way scrolling, windowing and data entry/display in screen forms. You can also write to screen many times faster than with BASIC and develop software to copy areas of the screen for cut and paste functions.

The multi-user **Data Base Management** module lets multiple users access the same data files across a local area network and still maintain data integrity. Users can build, add and delete indexes; reformat files and much more. Now BASIC users can control data as never before.

The **Network Interface modules** make it possible for BASIC programs to issue NETBIOS commands.

BASIC limits you to 64K of RAM for strings. **EXIM ToolKit's Memory Manager** lets you access up to **256K additional RAM** for strings. And, we offer free technical support, 7 days a week.

Order your **EXIM ToolKit** today. It's only $99.95, ($199.95 with Network Interface Modules) plus $5.00 shipping and handling. (No licensing or royalties required.) If within 30 days, you are not completely satisfied with the **EXIM ToolKit,** return it to your dealer for a complete refund.

The **EXIM ToolKit** is so comprehensive, we're convinced it's the only BASIC Software Library you'll ever need.

EXIM

EXIM SERVICES OF N.A. INC.

P.O. Box 5417
Clinton, New Jersey 08809
(201) 735-7640

Number 137 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The Resource Journal For IBM PC Programmers
PROGRAMMER'S Journal

DIRECT MAIL ORDER

ORIGINAL DESIGN T-SHIRTS

BORN TO PROGRAM PJT101

NOW! Soft Wear from Programmer's Journal!

Only $9.95 each, Postage paid. Classic PJ cartoons & cover graphics. Reproduced as dynamic Multi-color imprints on high quality 50/50 cotton T-Shirts. In sizes S, M, L, XL. Born to program? Send for yours today!

RS-232 HEAD PJT103

PARALLEL PORTS PJT102

COMMON C BUG PJT104

BACK ISSUES

Vol. 1.1: Norton batch files and Pascal I/O; Program style manual; COBOL on the IBM PC; Benchmarks; System analysis; BASIC modules; Rounding error.

Vol. 1.2: Defining character sets; C86 BIOS calls; Troubleshooting; Copy protection; Pascal code; Input routines; Writing an assembler.

Vol. 1.3: ASM dumb terminal; ASM tutorial routines; C source code; ASM graphics line drawing; Text editors reviewed; Utilities; Self-documenting programs; Text files to dBASE II; dBASE II PEEKS & POKES.

Vol.1.4: Diskette directory via Pascal; MASM WHILEWEND; C/ASM interface; ASM stay resident routine; Structured programming; Software law; Pascal tutorial; Default values and variable names.

Vol. 1.5: Multi-tasking software review; Transporting C programs; Programmer's calculator; dBASE II bargraphs; Utilities; ROM BIOS equates; Pascal check program; Combining modules in Assembler; Structured programming in BASIC; Agreement outline.

Vol. 1.6: ASM printer configuration; DOS patch utility; Hi-speed graphics; Software ownership; Structured programming revisited; C and BASIC compared; Software protection; dWINDOW review.

Vol. 2.1: Freelance consulting; Sorting in C; Pascal text editor source code; CRT distress; Compiled BASIC; Assembly modules for C; Designing a relational database; Ultra Utilities; Punctuation and Style reviewed.

Vol. 2.2: Interview with Brian Kernighan; Understanding C through assembly language; Utility routines in C and assembly; Music in C; Automatic line number optimizing in BASIC; Assembly language tutorial; Full-screen input routines in BASIC; dBASE II printer control codes; Pascal tools reviewed; C programming support products.

Vol. 2.3: Software development environments; Creating a productive screen handling tool; Assembly tutorial; Generating graphics within windows; Composition and editing system for musical ideas; Hybridizing assembly and BASIC; Creating screens; Linking Assembler to Pascal and BASIC.

Vol. 2.4: Writing a custom ROM BIOS for PC DOS; RISC, Transputers, and mass storage; The future of disk protection; 8088 assembly macros; Random access files in BASIC; C utility routines you can buy; Using the PMATE macro language; Choosing a database manager; E/SP syntax-checking; Pascal editor reviewed.

Vol. 2.5: Developing the APX CORE Executive; From assembler to C via a 21 byte .EXE main( ); Scroll control using BASIC and Assembler; Faster screen handling with BASIC and ASM, Random access files, Part 2; RTOOL, a PC-resident programmer's toolkit; dBkey reviewed; Pascal's pointer power.

Vol. 2.6: Periscope--building a hardware board for the IBM PC; Intel 80286 bugs by FNE; AT&T surprises all; Field Usage and pointer subtraction in C; Calendar functions for BASIC; Memory segmentation; Reduce your .EXE file sizes; Enhancing Debug; dBASE II to dBASE III? THINKTANK reviewed.

Vol. 3.1: November's Comdex; C--its uses and abuses; Using the C preprocessor; Memory segmentation examples; Index files, Part 1; Structured programming language; Periscope, a symbolic debugger.

Vol. 3.2: UNIX! (again); Artificial Intelligence; Doctor C's tips--pointers to functions; Calendar date routines in BASIC; Calculate record layouts, automatically!

Vol. 3.3: NAPLPS and the professional programmer; Instant-C; Printer control from a batch file; Repositioning the stars in Starfinder; Reducing EXE file size through data positioning; tools for better programming.

Vol. 3.4: More UNIX stuff; C programming books, Part 1; C code excerpts, Part 1; How to cure the "dreaded locked keyboard" condition; Calling MS DOS from Turbo Pascal; DOS 2.x sort and find filters; the SET Tools.

Vol. 3.5: Windows done here; C programming books, Part 2; C code excerpts, Part 2; A menu-driven PC using DISPLAY.COM; Turbo Pascal and MASM: making them work together; dBASE II routines; Turbo Pascal 3.0; TopView Programmer's Toolkit.

Vol. 3.6: About C's HLL overhead; How to recover all your costs; Report on the Comdex and Usenix shows; The Santa Cruz Operation XENIX for the IBM PC; Common C bugs; CTOOLS2; Turbo Pascal 3.0 and subdirectories; More dBASE II routines; Clipper reviewed; A better way to interface BASIC and Assembler.

Vol. 4.1: FNE on PC/AT hard disk problems and plunging 64K DRAM prices; AUTOINC explained; Sorting using an AVL tree; Combining binary trees; Using DOS to call programs from Turbo Pascal; An operating system for the people; Final installment of Brett Salter's dTOOLS; Two microcomputer graphics books reviewed.

Vol. 4.2: FNE on Intel 80386, single- vs. multi-user, multi-tasking computing, benchmarks; The home office syndrome; Frank Greco's industry news; Funky string functions; Capturing file handles in Turbo Pascal; Proposed ANSI C Standard, Part 1; Optimizing for speed in Assembler; Memory resident control over print screen functions; Structuring BASIC output with field statements; QuickBASIC Compiler and networking; Programming for Printer Output reviewed.

Vol. 4.3: FNEanother RISC machine, the paging game, software viewpoint, programmer talent, and Aloha, 32032; Sun Microsystems' next generation PC; Consulting—reusable modules; INT24.PAS, a critical error handler for Turbo Pascal; Memory model independence in Assembler; An approach to windowing for the assembly language programmer; Proposed ANSI C Standard, Part 2; Microsoft and Lattice C 3.0 reviewed: Accelerator boards for software development reviewed.

Vol. 4.4: Hal W. Hardenbergh on adult toys, tomorrow's micros, HAL-GOL update, benchmarks, and RISCs; Frank D. Greco on parallel computers, RISC, UNIX, Consulting—efficient custom programming; DOS 2 file handles and file I/O; Programming the WordStar Document File Conversion Utility; More Optimizing for Speed; Teach an old compiler a new trick; Memory resident programs in C; Tips on formatting with BASIC PRINT USING statements; the structure of dBASE II and III file headers.

Vol. 4.5: Hal W. Hardenbergh on 32-bit PCs, the 68040, Weitek's data-path part, etc.; Frank D. Greco on large screen monitors, language translators, UNIX, and more; Prolog intro, Turbo Prolog reviewed; Consultingtax traps, consulting styles, nest eggs; ASM—Search and Display; ASM—the GROUP statement; The proposed ANSI C Standard, Part 3; C—measuring stack usage and Lattice C's startup code; Short boards reviewed.

Vol. 4.6: DTACK GROUNDED; Unix Books, Part 1; How many amoebas at one o'clock; DOS functions 45H and 46H; DOS Terminate ans Stay Resident; Turbo Pascal inline code; Pascal—Assembler souce code for DOS function calls; Alignment problems in C and Managing memory allocation; Proposed ANSI C Language Standard, Part 4; The Canon A1 laser printer; Genifer dBASE applications generator reviewed; Advanced Trace86 debugger reviewed.

Vol. 5.1: DTACK GROUNDED; The new DOS; Marketing and rules to live by; File redirection; Inside the EGA; String allocation in Pascal; ANSI C Language Standard, Part 5; More tips on formatting BASIC data.

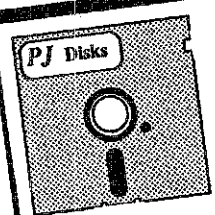
Vol. 5.2: DTACK GROUNDED; Starting your own software house; Pink noise; TSR graphics print screen utility; Coding guidelines for resident programs; Parallel processing with the EGA; C—portability in practice and using header files; Proposed ANSI C Language Standard, Part 6: Indexing in BASIC; A first look at Turbo C.

Vol. 5.3: Starting your own software house--publicity; Unix books, Part 2; Consulting—merger mania; EGA data control; Modula 2 for Pascal lovers; Optimizing Turbo Pascal; ANSI C language standard; Coping with ANSI C and forcing file updates; Formatting BASIC output with buffered PRINT USING.

Vol. 5.4: Starting your own software house—technical support; Unix books, Part 3; Consultant productivity and effectiveness; EGA set/reset capabilities; ANSI C language standard; Character tables and string functions in C; MWWIND—a menu with windows utility; Variable passing and hiding in QuickBASIC 2 subprograms.

Vol. 5.5: Intel's 80386 vs. OS/2; Metagraphics President Jack Davis interview; Consulting—making computer tools pay for themselves; ASM—the invisible CapsLock; TSRs using expanded memory with Turbo Pascal; Note on optimizing Turbo Pascal; Proposed ANSI C Language Standard; Printer output from Turbo C programs; Matchup of Turbo Basic 1.0 and QuickBASIC 3.0.

To order, just use the convenient form located on the perf insert, at right. If the card is missing, you may place credit card orders by calling our office at:
(503) 747-0800



NOW! PJ CODE ON DISK—

Brett Salter's dBASE II Routines *from 4.5, 4.6 & 5.1* **$10.00**

ALSO—

Michael Abrash EGA Code *from 5.1, 5.3 & 5* **$10.00**

PRODUCED 2004 BY OSZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Attention Programmers!
NEW TURBO C VERSION

HERE'S THE QUICK, EASY WAY TO MAKE SCREENS, DEMOS AND TERRIFIC PROGRAMS!

DisplayExpress™: A fast, efficient and easy-to-use screen manager.

DisplayExpress™ helps you make efficient screens as easily as you would type a letter. Such screens take little space (less than 2 kbytes in the average) and are displayed instantaneously.

Colors...Character graphics...Editing power...Automatic drawing commands...And a very compact run-time support: 6 kbytes only, yet it includes all you need to load and display screens and menus!

The package also includes the DemoExpress® Viewer which allows you to make demodiskettes and slides quickly and easily. Ideal for your presentations, tutorials and continuously running demos.

C-Display Librarian™: The professional programmers' choice.

C-Display Librarian™ provides complete support for the man-machine interface. Perfoms BIOS video, screen editing, data entry and validation, keyboard control and bit mapped graphics.

Modular design...Small size...Great speed... Professional documentation...Full source... Compatible with Microsoft and Lattice compilers. Available for IBM Personal System/2 or PC/XT/AT.

30-day money-back-back guarantee!
No royalties on programs or demos!
Free programming examples with source!
Free shipping within the United States on all prepaid orders!

"If you want to include bit-mapped graphics in your application, then this might be the package for you"
(D. Dramheller and M. Wilson, Computer Language)

"DisplayExpress and C-Display Librarian are very fine products and I am looking forward to incorporating them into my applications development"
(R.W. Hirn, Vice-President, NHDS Inc., Chicago, IL)

"DisplayExpress and C-Display Librarian are a great time saver and should be in the tool-kit of every C language programmer"
(D. Wilson, McLean, VA)

YES!

I want to make screens, demos and terrific programs right now!

Please rush me:

☐ **DisplayExpress**™ $69
☐ **C-Display Librarian**™ $69

Sales tax (N.Y. residents only) $ _____

TOTAL $ _____

ALL FOREIGN ORDERS: add $10.00 per product and make payment by bank draft payable in US dollars and drawn on a US bank.

NAME _____

COMPANY _____

TEL.: _____

ADDRESS _____

CITY _____

STATE _____ ZIP _____

COUNTRY _____

New! Low Price!

DisplayExpress™ $69
C-Display Librarian™ $69

SYDETECH
SYSTEM DEVELOPMENT TECHNOLOGIES INC.
43-23 Colden Street, #17C • Flushing, New York 11355

For COD orders or the name of the nearest dealer please call (718) 886-0221

Number 74 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

THE MICRO TECHNICAL JOURNAL
MICRO CORNUCOPIA
The Way We Were...
PC Mouse Drivers page 3
Build A Midi Interface For Your PC page 14
Designing A Database, Part 2 page 20
Interrupts On The PC page 30
Hacker's View Of MS-DOS Vrs 3.X page 46
And Much Much More
Mouse Control

CALL 1-800-888-8087 AND SUBSCRIBE TODAY!

If you're looking for real hardware and software projects instead of reviews, reviews, reviews! Looking for candid discussions of who's ROM BIOS (or hard drive) works and whose doesn't; then call us. Join the group.

We're bigger, better, funnier, broader, snappier, and snappier than ever. We're independent and proud of it. The "Mother Earth News™" of the computer world.

It's the most (computer) fun you'll ever have. We guarantee it.

Here are just a few of the articles in the September/October issue:

- Desktop publishing on a PC.
- Part 3 of the database design series (starring Dr. Dobbs, the famous veterinarian and Dee Base, a most comely consultant).
- Controlling 110V AC from your PC's printer port. This is part of a design-it and build-it series by Bruce Eckel.
- How to write software which detects and uses expanded memory on PCs and clones.
- When you should (and shouldn't) market your software as shareware (by folks who are making close to a $1,000,000 per year selling their own programs as shareware).
- How to build a professional quality graphics scanner for $6.00.

And these are just the high-lights from one issue.

YES, I WANT TO SUBSCRIBE!

☐ New ☐ Renewal

☐ 1 year (6 issues) $18
☐ 2 years (12 issues) $34
☐ 3 years (18 issues) $48

SAVE 24% OFF Newstand Price
(Price for U.S. Only)

☐ VISA ☐ CHECK ENCLOSED
☐ MASTERCARD ☐ BILL ME

☐☐☐☐–☐☐☐☐–☐☐☐☐–☐☐☐☐–☐☐☐☐ EXP. DATE

NAME

COMPANY

ADDRESS

CITY

STATE ZIP

MAIL TO: Micro Cornucopia/P.O. Box 223/Bend, Oregon 97709

1-800-888-8087 Monday thru Friday 9-5 Pacific Time

Number 119 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Microway has released the mW1167 numeric coprocessor card containing the Weitek chip set for these new 386 motherboards. The 121 pin grid array is a superset of the 80387 pin grid and also supports an 80387 chip. The new motherboards currently supporting the Weitek chip set are AT&T 6386, Tandy 4000, and a newly announced Compaq. Microway is also shipping the NDP FORTRAN-386 compiler, which supports the Weitek chip set as well as the 80387. These compilers produce 32-bit 80386 native code and run under Unix 5.3 or MS-DOS as extended by Phar Lap. The FORTRAN compiler takes full advantage of the 4 gigabyte address space of the 80386. C and Pascal 386 compilers will soon follow. The FORTRAN compiler is priced at $595.

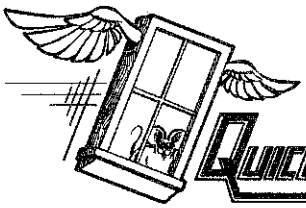
Concurrent DOS
Digital Research
Box DRI
Monterey, CA 93942
800/443-4200
386 3 user $395, 10 user $495
XM 3 user $295, 6 user $395

Digital Research has begun retail distribution of Concurrent DOS 386 Version 1.1 (CDOS 386), an enhanced multi-user, multitasking operating system for 80386 machines. The retail product is sold in a 3 user or a 10 user configuration. The key features of this new release are ease of installation, familiar DOS command structure, on-board memory paging (no device drivers or special boards are required), runs all major DOS applications, and can access up to 4,000 megabytes of system memory. CDOS 386 is supported on the Compaq 386 and 100% compatibles.

DRI also released an enhanced version of Concurrent DOS XM, Release 5.2, for the PC XT/AT. This new version offers improved EGA support (16 colors), enhanced keyboard support, up to four disk partitions per hard disk, and support of the AST-Four Port/DOS card and the four-port Hostess Multiport Network Adapter. CDOS XM supports conventional and/or expanded memory using memory cards supporting the EEMS specification (AST Research). CDOS will run most major DOS applications. Upgrades from Version 5.0 are available at no charge while earlier versions (CDOS 4.1) may be upgraded for $95.

SAVE HOURS OF VALUABLE PROGRAMMING TIME!

FREE Demo Disk



Quick Windows

A low cost Window Management Library for BASIC

- Create windows, mouse/kb driven pop-up and pull-down menus, and powerful data entry routines quickly and easily.
- Fast! Written entirely in assembly language
- Efficient! Uses less than 21K of object code
- Full mouse interface for Microsoft compatible mouse.
- Supports MDPA, CGA, and EGA automatically.

Comes complete with a library of over 60 functions, a fully-documented manual, many programming examples, and a demo/tutorial program to show you how to use each of the functions. All for only $79. Add $20 for the QuickWindows™ assembly language source code.

No royalties are required for using QuickWindows™ in your programs and not copy protected. 30-day money back guarantee. For IBM/True compatibles and Microsoft QuickBasic or BASCOM, DOS 2.0 or later.

Call now for your FREE demo disk!
401-274-5465
Visa, MC and Phone orders welcome.

Dealer discounts available.

Software Interphase INCORPORATED
5 Bradley Street, Suite 7
Providence, RI 02908-2304

Number 115 on the Reader Service Card

Events

Forth National Convention
Forth Interest Group
P.O. Box 8231
San Jose, CA 95155
408/277-0668

The Forth Interest Group (FIG) will be holding their 9th Annual Forth National Convention and Exhibit on November 13–14 at the Red Lion Inn in San Jose, California. The program theme this year is the Evolution of Forth. Conference sessions and technical workshops are scheduled along with vendor exhibits. Registration is $20 in advance, $25 at the door if you need to get some new threads

X Windows Correction

MIT Microcomputer Lab
77 Massachusetts Avenue
Cambridge, MA 02139
617/253-7185

Over the summer MIT changed the pricing and availability of X Windows, the public domain network windowing protocol we talked about in Volume 5.5. The MIT phone number for information and ordering also changed. The documentation on X Windows is now available separately only to MIT students. The cost of the total distribution (written documentation and sources for various machines) available on 9 track magnetic tape in Unix format has been increased to $150. The entire distribution (20+ files) plus documentation has also been placed on ARPANET and is accessible using anonymous ftp. On the East Coast, access is at zap.mit.edu, and on the West Coast at decwrl.dec.com. The X Windows material should be in a /pub directory. We hope someone will download the X Windows distribution to a more accessible location for the PC community. ♦

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

DOS User's Group

P.O. Box 26601

Las Vegas, NV 89126

Annual Membership
$25.00 (U.S.)
$35.00 (Foreign)

Quarterly Newsletter containing:

DOS Command Overview & Tutorials
DOS Customization Hints
DOS Question & Answer Section
OS/2 Commands & Concepts
OS/2 Trends & Impacts
OS/2 Question & Answer Section
MS Windows In Detail
Third Party Product Reviews

Number 128 on the Reader Service Card

U.S. Postal Service
STATEMENT OF OWNERSHIP, MANAGEMENT AND CIRCULATION
Required by 39 U.S.C. 3685)

1A. TITLE OF PUBLICATION	1B. PUBLICATION NO.	2. DATE OF FILING
Programmer's Journal	0 7 4 7 5 8 6 1	10/1/1987
3. FREQUENCY OF ISSUE	3A. NO. OF ISSUES PUBLISHED ANNUALLY	3B. ANNUAL SUBSCRIPTION PRICE
bi-monthly	6	$24.00

4. COMPLETE MAILING ADDRESS OF KNOWN OFFICE OF PUBLICATION (Street, City, County, State and ZIP Code) (Not printers)
150 North 4th Street, Springfield, OR 97477

5. COMPLETE MAILING ADDRESS OF THE HEADQUARTERS OF GENERAL BUSINESS OFFICES OF THE PUBLISHER (Not printer)
150 North 4th Street, Springfield, OR 97477

6. FULL NAMES AND COMPLETE MAILING ADDRESS OF PUBLISHER, EDITOR, AND MANAGING EDITOR (This item MUST NOT be blank)

PUBLISHER (Name and Complete Mailing Address)
Elizabeth Oakley, 150 North 4th Street, Springfield, OR 97477

EDITOR (Name and Complete Mailing Address)
Charles Aylworth, 150 North 4th Street, Springfield, OR 97477
Erin O'Connor, 150 North 4th Street, Springfield, OR 97477

MANAGING EDITOR (Name and Complete Mailing Address)

7. OWNER (If owned by a corporation, its name and address must be stated and also immediately thereunder the names and addresses of stockholders owning or holding 1 percent or more of total amount of stock. If not owned by a corporation, the names and addresses of the individual owners must be given. If owned by a partnership or other unincorporated firm, its name and address, as well as that of each individual must be given. If the publication is published by a nonprofit organization, its name and address must be stated.) (Item must be completed.)

FULL NAME	COMPLETE MAILING ADDRESS
Oakley Publishing Co.	150 N. 4th St., Springfield, OR 97477
Elizabeth Oakley	150 N. 4th St., Springfield, OR 97477
James Oakley	150 N. 4th St., Springfield, OR 97477

8. KNOWN BONDHOLDERS, MORTGAGEES, AND OTHER SECURITY HOLDERS OWNING OR HOLDING 1 PERCENT OR MORE OF TOTAL AMOUNT OF BONDS, MORTGAGES OR OTHER SECURITIES (If there are none, so state)

FULL NAME	COMPLETE MAILING ADDRESS
none	

9. FOR COMPLETION BY NONPROFIT ORGANIZATIONS AUTHORIZED TO MAIL AT SPECIAL RATES (Section 423.12 DMM only)
The purpose, function, and nonprofit status of this organization and the exempt status for Federal income tax purposes (Check one)

[X] HAS NOT CHANGED DURING PRECEDING 12 MONTHS
[ ] HAS CHANGED DURING PRECEDING 12 MONTHS (If changed, publisher must submit explanation of change with this statement.)

10. EXTENT AND NATURE OF CIRCULATION	AVERAGE NO. COPIES EACH ISSUE DURING PRECEDING 12 MONTHS	ACTUAL NO. COPIES OF SINGLE ISSUE PUBLISHED NEAREST TO FILING DATE
A. TOTAL NO. COPIES (Net Press Run)	13,000	17,000
B. PAID CIRCULATION 1. Sales through dealers and carriers, street vendors and counter sales	4,844	7,984
2. Mail Subscription	5,477	5,600
C. TOTAL PAID CIRCULATION (Sum of 10B1 and 10B2)	10,321	13,584
D. FREE DISTRIBUTION BY MAIL, CARRIER OR OTHER MEANS SAMPLES, COMPLIMENTARY, AND OTHER FREE COPIES	463	511
E. TOTAL DISTRIBUTION (Sum of C and D)	10,784	14,095
F. COPIES NOT DISTRIBUTED 1. Office use, left over, unaccounted, spoiled after printing	1,002	2,902
2. Return from News Agents	1,214	3
G. TOTAL (Sum of E, F1 and 2--should equal net press run shown in A)	13,000	17,000

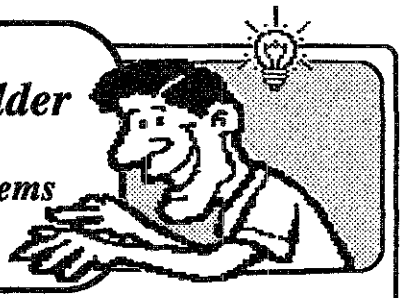
11. I certify that the statements made by me above are correct and complete

SIGNATURE AND TITLE OF EDITOR, PUBLISHER, BUSINESS MANAGER, OR OWNER
Elizabeth Oakley
Publisher

PS Form 3526 (See instruction on reverse)

You've Seen all the Rest
Now go for the Best!

ScreenBuilder
from
Gilmore Systems



• Absolutely NO Royalties!
• Create Interactive, Animated, Automated:
  • Demos
  • Tutorials
  • User/Program Shells
  • Teaching Aids
  • Children's Education
  • Continuously Running Presentations
• Orders of Magnitude above the rest:
  • Scund and Music Capabilities
  • Mathematics & String Logic
  • User inputs and outputs
  • Printer Control
  • Loops, Subroutines, Decision Making
  • Screens are directly useable by application programs
  • Supports all 256 colors & characters
  • Screen Capture Program
  • Runs on CGA, EGA and Monochrome
  • Much, Much More!
• Easy to use. Has context sensitive help.
  • Complete, comprehensive 150+ page User Manual guides you through every step of the way.

Just $59.95
Complete!

Sample Demo Disk available for $2.00
(Credited towards your first order)

Call Today to order by credit card
or send $59.95 plus $3.00 shipping/handling to:

GILMORE SYSTEMS
P. O. Box 3831
Beverly Hills, CA 90212-0831
VISA (213) 275-8006 MasterCard

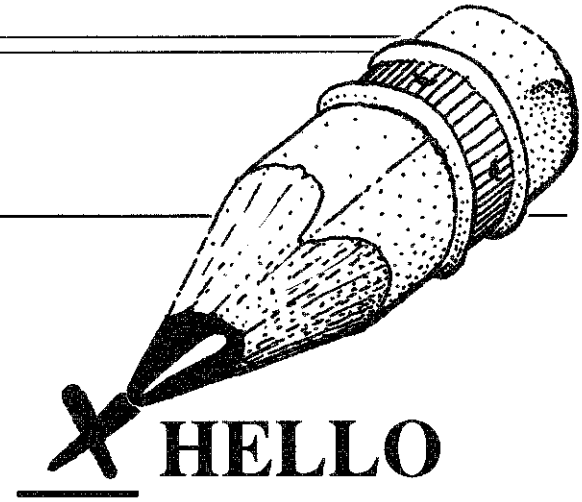
Requires IBM PC, XT, AT or compatible. DOS 2.0 or higher, 320 K RAM minimum. Fixed disk recommended.

Number 135 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

SURVEY

Please help us out. We want to know more about you. Our editors want to know what you'd like to find in the pages of Programmer's Journal, and our advertisers have their marketing preoccupations. Take a few minutes to fill out this survey and send it back to us postage paid. Thanks for your help.



Do you own a microcomputer? ___ YES ___ NO

___ IBM	___ Compatible	Non-IBM standard	
___ PC	___ 80386	___ 68X00	___ Mini
___ XT	___ Mini	___ Z-80	___ Mainframe
___ AT	___ Mainframe	___ Other (Specify)	

What hardware do you plan to buy for yourself in the next twelve months?

___ Color Monitor	___ Memory Board	___ Expansion Board
___ Hard Disk	___ Tape Backup	___ Accelerator Board
___ Modem	___ 386 Motherboard	___ Laser Printer
___ CGA Board	___ EGA Board	___ VGA Board
___ PC (Specify)		

What software do you plan to buy for yourself in the next twelve months?

___ Compiler	___ Debugger	___ Screen Editor
___ Operating System	___ DOS Shell	___ Disk Utility
___ Data Base	___ Language	___ Desktop Publishing
___ Other (Specify)		

How many years have you been using microcomputers?

___ 0-1 ___ 1-5 ___ 5-10 ___ 10+

What do you use a microcomputer for? (Check all answers that apply.)

__ Software Development	__ Education	__ Personal/Financial
__ Word Processing	__ Spreadsheet	__ Accounting
__ Project Management	__ Process Control	__ Telecommunications
__ Data Base Management	__ Statistical Analysis	__ Tax Calculation
__ Portfolio Management	__ Graphics/Charts	__ Science & Engineering
__ Other (Specify)		

Do you use a mainframe? ___ YES ___ NO

Do you use a mini computer? ___ YES ___ NO

How many years have you been programming?

___ 0-1 ___ 1-5 ___ 5-10 ___ 10+

How many years have you been a microcomputer programmer?

___ 0-1 ___ 1-5 ___ 5-10 ___ 10+

How do you rate your programming skills?

___ Minimal ___ Medium ___ Professional

What programming languages do you use? (Check ones used most often.)

__ Ada	__ FORTH	__ PostScript	__ APL	__ FORTRAN
__ PROLOG	__ Assembly	__ LISP	__ Rbase	__ BASIC
__ C	__ LOGO	__ REXX	__ Lotus	__ RPG
__ C++	__ Modula-2	__ Smalltalk	__ COBOL	__ Pascal
__ SNOBOL	__ dBASE	__ PL/I	__ SQL	__ Other

Which one is your favorite?

What programming languages that you don't use now would you like to learn? (Check all answers that apply.)

__ Ada	__ FORTH	__ PostScript	__ APL	__ FORTRAN
__ PROLOG	__ Assembly	__ LISP	__ Rbase	__ BASIC
__ C	__ LOGO	__ REXX	__ Lotus	__ RPG
__ C++	__ Modula-2	__ Smalltalk	__ COBOL	__ Pascal
__ SNOBOL	__ dBASE	__ PL/I	__ SQL	
__ Others(specify)				

What computer-related magazines do you read regularly? (Check appropriate box.)

	Subscription	Newsstand	Bookstore	Library	Friend/Colleague
AI Expert					
AI Magazine					
BYTE					
The C Journal					
Comm. of the ACM					
ProFiles					
Computer Language					
Computerworld					
CP/M Users Guide					
Dr. Dobb's Journal					
Electronic News					
IEEE Expert					
IEEE Spectrum					
Infoworld					
MacWorld					
Micro Cornucopia					
Micro/Systems Journal					
Mini Micro Systems					
PC					
PC Tech Journal					
PC Week					
PC World					
Publish!					
SigPlan Notices					
Systems & Software					
UNIX Review					
UNIX World					
Others (Specify)					

How long have you been reading ***Programmer's Journal***?

___ 0-1 year ___ 2 years ___ 3 years ___ 4 years ___ 5 years

How do you get your copy of ***Programmer's Journal***?

___ Sub ___ Friend ___ Newsstand ___ Library

How many people besides you read your copy of ***Programmer's Journal***?

___ 1 ___ 2 ___ 3 ___ 4 ___ 5 or more

Do you keep your ***PJ*** for reference? ___ YES ___ NO

Do you use the code listings you find in ***PJ*** in your own programming?

___ Never ___ Occasionally ___ Often

Would you like to be able to access code listings that appear in ***PJ***?

___ YES ___ NO

If YES, via what medium?

___ Disk ___ Bulletin Board ___ Cauzin SoftStrip™

(Over)

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

What hardware/operating system do you use at work?

What hardware/operating system do you use at home?

Do you design and/or write software as an employee?
___ YES ___ NO

If YES, check all answers that apply:
__ Commercial distribution __ Consulting clients __ Internal company use

Do you design and/or write software as an independent?
___ YES ___ NO

If YES, check all answers that apply:
__ Commercial distribution __ Consulting clients __ Personal use

Are you developing software for OS/2? ___ YES ___ NO

What is the primary business activity of your firm?

__ Software Development
__ Mining or Construction
__ Transportation
__ Computer-related Retailing
__ Other Retailing/Wholesaling
__ Health/Medical Services
__ Education
__ Other (Specify.)
__ Computer Equip. Mfg.
__ Application Software
__ Communications
__ Finance,Banking
__ Insurance
__ Government
__ Publishing
__ Other Mfg.
__ Agriculture
__ Utilities
__ Real Estate
__ Legal Services
__ Data Processing
__ Entertainment

What is your job title?

What best describes the work you do? (Check all answers that apply.)

__ Pres./Chrmn./Partner
__ Owner/CEO
__ Sales/Marketing
__ Programmer/Analyst
__ Software Engineer
__ Other (Specify.)
__ Admin./Ops./Plant
__ Gen'l. Mgr.
__ R&D/Engineering
__ Purchasing
__ Consultant
__ Educator
__ Production Mgr.
__ DP/MIS
__ Communications
__ Software Entrepreneur

How many people work for your company?
___ 1 ___ 10 - 49 ___ 50 - 99 ___ 100 +

Do you advise your company on purchases of microcomputers and products?
___ YES ___ NO

How much does your company plan to spend in 1988 on such products?
___ $0-1,000 ___ $1,000-$5,000 ___ $5,000-$10,000 ___ $10,000 +

How much do you plan to spend personally in 1988 for such products?
___ $0-1,000 ___ $1,000-$5,000 ___ $5,000-$10,000 ___ $10,000 +

What is your combined household income?

___ $00,000-$10,000
___ $10,001-$20,000
___ $20,001-$30,000
___ $30,001-$40,000
___ $40,001- $80,000
___ $80,000 +

What is your individual earned income?

___ $00,000-$10,000
___ $10,001-$20,000
___ $20,001-$30,000
___ $30,001-$40,000
___ $40,001- $80,000
___ $80,000 +

Are you male or female? ___ M ___ F

What is your age group? (Circle one)
__ 18-24 __ 25-34 __ 35-44 __ 45-54 __ 50-54 __ 55+

What is your education?

___ High School
___ College 1 2 3 4
___ Bachelor's
___ Master's
___ Ph.D

What is the main reason you read ***Programmer's Journal***?
__ Languages __ Code __ News __ Cartoons __ Business Advice

What do you like best about ***Programmer's Journal***?

What do you like least about ***Programmer's Journal***?

What articles / features would you like to see in future issues of ***PJ?***

Fold Here

Fold Here

Name_____

Company_____

Address_____

City_____

State_____ Zip _____

Phone#_____

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL
FIRST-CLASS MAIL PERMIT NO. 44 EUGENE, OR

POSTAGE WILL BE PAID BY ADDRESSEE

PROGRAMMER'S JOURNAL
P. O. BOX 30160
Eugene, OR 97403

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Tape Closed

CODE IN THIS ISSUE

The code listings in this issue of ***Programmer's Journal*** are available on:

FIDO/OPUS network
Blazer Net 152/9
503-343-9724
and
Turbo Source Search BBS
P.O. Box 876
Scituate, MA 02066

617-545-9131 *RS#35*

(See ad on page 59)

THE MEMORY RESIDENT HELP SYSTEM AND DEMO PROGRAM GENERATOR

Custom fit this flashy, menu driven help system to any new or old program without programming, instantly! With the Demo Program Generator, create **LIVE DEMOS** for any program. Not just another demo program that runs a picture show.

Actually run your applications **LIVE** through the demo windows with ***The Pointer. Rush $129.95 U.S. to:***

Stau Enterprises Inc.
3650 Kaneff Cres., Suite 1808
Miss. Ont. Can. L5A 4A1
416-270-6673 *RS# 134*

HOW COMPLEX IS YOUR SOFTWARE?

Find out with PC-METRIC. PC-METRIC can help you write programs that are easier to understand and maintain. Computes many popular complexity metrics. Versions for C, Modula-2 and Pascal.

Demo disk $10

SET Laboratories, Inc.
P. O. Box 03627
Portland, OR 97203
503-289-4758 *RS#149*

TURBO PASCAL PROGRAMMERS

The SOURCE CODE for Turbo Pascal is now available! For a remarkably low price users can now have fully commented source code for Turbo Pascal v3.01A (binary arithmetic). This includes the run-time library, editor and compiler. Everything you need is included, even an assembler! Free info.

Price: IBM-PC $67.50, Z-80 $45 (specify disk required)

C.C. Software
1907 Alvarado Ave. #B
Walnut Creek, CA 94596
415-939-8153 *RS# 132*

MODULA-2 COMPILER

Generates native code for LARGE and HUGE memory models. Integrated editor and make. Intelligent linker. Inline assembler. Coroutines, IOTRANSFER Requires PC or clone with 512K, better with 640K.

Distributed as Shareware, registration is $35. Download an evaluation copy from our BBS at 404-497-0931, or order from:

Fitted Software Tools
P.O. Box 956129
Duluth, GA 30136 *RS# 138*

INSTANT ACCESS

To Any Directory without typing full path names. The COMMAND System automatically generates Short Names for each directory program. Fast Programs; functions DOS forgot. Super Directory program. Fantabulous SEARCH Program. 30-day Money-Back Guarantee.

Buy Now: Introductory Price of **$75.** Send check or call with VISA or MC order. Free U.S. shipping.

CompuMagic, Inc.
P.O. Box 437
Severn, MD 21144
301-969-8068 *RS# 133*

CONVERT BASIC TO C OR PASCAL

BAS_C™ converts BASICA to C which can be directly compiled by any C compiler. BAS_PAS™ converts BASICA to Turbo Pascal™. Both BAS_C™ and BAS_PAS™ transform the spaghetti code to the scoped, indented structured program automatically. Deadcode. Graphic, Random file.

Sample Conversion: Send your own BASIC program (1 line / 1¢) and receive the printout.

Demo disk $5.00

Gotoless Conversion
P.O. Box 835910
Richardson, TX 75083
214-404-1404 *RS#127*

COMMUNICATIONS

Custom solutions for Asynchronous, Bisynch, SDLC, Multiline, Multitasking. Unattended Remote Polling Package with Scheduler, Mailbox, Remote Batch and Full Duplex Protocol. Many communications utilities.

ModemArt
P.O. Box 0081
Richardson, Tx 75083-0081
214-530-9572 *RS#130*

This publication is available in microform.

University Microfilms International reproduces this publication in microform: microfiche and 16mm or 35mm film. For information about this publication or any of the more than 13,000 titles we offer, complete and mail the coupon to: University Microfilms International, 300 N. Zeeb Road, Ann Arbor, MI 48106. Call us toll-free for an immediate response: 800-521-3044. Or call collect in Michigan, Alaska and Hawaii: 313-761-4700.

☐ Please send information about these titles:

Name _____

Company/Institution _____

Address _____

City _____

State _____ Zip _____

Phone ( ) _____

University Microfilms International

Number **152** on the Reader Service Card

LAHEY SETS NEW FORTRAN STANDARDS

LAHEY PERSONAL FORTRAN 77 - The full ANSI 77 Standard and Debugger for $95.00.

F77L - "EDITOR'S CHOICE" *PC Magazine*
"...the most robust compiler tested." *Micro/Systems Journal*
"...the ultimate fully-blown ANSI FORTRAN 77... a fantastic product." *PC Australia*

F77L-EM - Breaks the DOS 640K program barrier with extended memory. The most powerful PC FORTRAN available.

FORTRAN TOOLS: Profiler, Mathematical Functions Library, Overlay Linker, Utility Libraries, Widows Library, Toolkit.

The difficult decision is not from whom you should buy your PC FORTRAN, but which Lahey compiler you should purchase.

Call us today to discuss your PC FORTRAN needs.

FOR INFORMATION OR TO ORDER:
1-800-548-4778
30-Day Money-Back Satisfaction Guarantee
Lahey Computer Systems, Inc., Box 6091, Incline Village, NV 89450 Tel: 702-831-2500 TLX: 9102401256

Lahey Computer Systems Inc.

Number **44** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

ADVERTISER INDEX

Reader Service#	Advertiser Name	Page Number
151	American Software International	45
86	Aspen Systems	31
30	Austin Code Works	61
51	Az-Tech Software, Inc.	59
2	Blaise Computing	3
3	Borland International	C4
144	*C Gazette*	71
38	C Ware Corporation	24
132	C.C. Software	79
133	Compumagic, Inc.	79
56	CompuView Products, Inc.	9
*	Creative Programming Consultants	64
83	Crescent Software	8
124	Custom Real-Time Software	68
5	Dair Computer Systems	43
128	DOS Users Group	76
113	ELFSOFT	13
70	E.C.I.	45
8	Ecosoft, Inc.	1
137	Exim Services of N.A.	71
138	Fitted Software Tools	79
118	Fleming Software	71
135	Gilmore Systems	76
*	Gimpel Software	65
127	Gotoless Conversion	79
122	Hauppauge Computer Works, Inc.	37
57	Hawkeye Grafix Inc.	51
139	I.M.S.I.	C3
63	Laboratory Microsystems, Inc.	45
44	Lahey Computer Systems	79
92	Manx Software Systems	19
140	Marshal Language Systems	29
141	Merlin Publishing Group	13
143	Metagraphics Software Corporation	22
45	MetaWare, Inc.	43
119	*Micro Cornucopia*	74
*	PLEASE CONTACT THESE ADVERTISERS DIRECTLY	

Reader Service#	Advertiser Name	Page Number
13	Micromotion	69
104	MicroWay	53
121	MISOSYS Inc.	23
94	MMC AD Systems	68
130	ModemArt	79
109	NanoSoft	70
16	Opt-Tech Data Processing	51
102	Oregon Software	59
145	Pathfinder Associates	18
80	Paul Mace Software	7
6	Periscope Company, The	66
146	Programmer's Connection	39-41
*	*Programmer's Journal*	72
*	Quaid Software Limited	43
97	Qualstar Corporation	51
147	Quarterdeck Office Systems	C2
46	Quilt Computing	31
*	Raima Corporation	65
148	SemWare	69
149	SET Laboratories, Inc.	79
98	Software Family, The	5
108	Software Garden	17
115	Software Interphase, Inc.	75
110	Softway, Inc.	31
47	Solution Systems	25
134	Stau Enterprises Inc.	13
85	Sunny Hill Software	67
74	Sydetech, Inc.	73
150	Synergy Systems	24
126	Tangent Designs, Inc.	49
117	Tom Rettig Associates	49
35	Turbo Source Search	59
152	University Microfilms International	79
125	V-Communications	12
73	Whitewater Group, The	57
123	Xenosoft	31
*	PLEASE CONTACT THESE ADVERTISERS DIRECTLY	

Articles Coming in Jan/Feb 1988, Volume 6.1

Write Mode 3 of the VGA
by Michael Abrash

Porting Mainframe Software to the PC
by M. Steven Baker

The software approach (focusing on FORTRAN compilers) and a hardware alternative

EMS 4.0—The New Spec
by John J. Newlin

Hell-for-Leather Turbo
by Jon Greenblatt

ADVERTISE IN
Programmer's Journal
January/February 1988
Volume 6.1
GALA 5th Aniversary Issue!
Space Reservation Deadline
Nov. 13, 1987
Materials due
Nov. 20, 1987
Call the *Programmer's Journal*
Advertising Department Today to
Reserve Your Space!
(503) 747-0800

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

TURBO PASCAL AND TURBO C... MEET TURBOHALO

"Ideal! TurboHALO does the job comparable to packages costing $3000 to $4000."
Jim Bromley
Superintendent of Spectrum Management

"TurboHALO is so fun... I use it to design programs as a hobby...It's got lots of ability."
William Porter
Control Systems Manager

"I like the speed of TurboHALO...it's ten times faster than the competition."
Deniz Terry
Doctoral Candidate

"We evaluated all of the graphic development packages for Turbo Pascal, and TurboHALO was the hands down winner!"
Quinn Curtis
Largest New England Distributor

It's time to put graphics into your programming.
A picture's worth a thousand words. So your programming isn't complete until you have graphics. TurboHALO brings your screen and printer to life with subroutines that draw, chart, map, and display. All with color, shape, clarity, perspective and motion. With TurboHALO, create any picture you can imagine.

TurboHALO gives you graphics power.
TurboHALO gives you everything you need for Turbo C and Turbo Pascal graphics programming. A library of over 150 graphics subroutines. Drivers for over 42 graphics hardware devices for the IBM PC family and compatibles. You can create the images you want, on the hardware you have!

Fast, proven and reliable.
TurboHALO is up to ten times faster than other graphics toolkits. And with TurboHALO you get proven, reliable programming tools used by professionals for years.

You'll like TurboHALO or your money back!
TurboHALO is available for only $95. You get an unconditional 45-day money-back guarantee on TurboHALO. To order, call your dealer or IMSI at (415) 454-7101 or (800) 222-4723; in CA (800) 562-4723; in Washington DC (202) 363-9340 or in NC(919) 854-4674.

YES, *I want to see the difference TurboHALO makes in my graphics programming! Rush me the following TurboHALO Graphics Toolkit(s) @ $95 each plus $3 shipping. California residents add 6% sales tax.*

☐ *TurboHALO for Turbo Pascal*
☐ *TurboHALO for Turbo C*
☐ *Check enclosed for $ _____ (made payable to IMSI)*
☐ *Charge my credit card for $ _____* ☐ *VISA* ☐ *MasterCard*

Signature
Card Number *Expiration Date*
Name
Title
Company
Street
City *State* *Zip*

IMSI™

Mail to:
IMSI, 1299 Fourth Street, San Rafael, CA 94901

PJ1187

FOR GRAPHICS PROGRAMMING

TurboHALO requires 256K memory (min); memory resident drivers require 2K (Turbo Pascal only); DOS version 2.0 or higher; Borland language/compiler required. TurboHALO is a trademark of Media Cybernetics and IMSI. Turbo C and Turbo Pascal are trademarks of Borland.

Number **139** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Program in the fast lane with Borland's new Turbo Pascal 4.0!

Our new Turbo Pascal® 4.0 is so fast, it's almost reckless. How fast? Better than 27,000 lines of code per minute.* That's more than twice as fast as Turbo Pascal 3.0.

4.0 Technical Highlights:

- Compiles 27,000 lines per minute
- Includes automatic project Make
- Supports > 64K programs
- Uses units for separate compilation
- Integrated development environment
- Interactive error detection/location
- Includes a command line version of the compiler
- Highly compatible with 3.0

For the IBM PS/2™ and the IBM® and Compaq® families of personal computers and all 100% compatibles

4.0 breaks the code barrier

No more swapping code in and out to beat the 64K code barrier. Designed for large programs, Turbo Pascal 4.0 lets you use every byte of memory in your computer.

4.0 uses logical units for separate compilation

Pascal 4.0 lets you break up the code gang into "units," or "chunks." These logical modules can be worked with swiftly and separately. 4.0 also includes an automatic project Make.

4.0's cursor automatically lands on any trouble spot

4.0's interactive error detection and location means that the cursor automatically lands where the error is. While you're compiling or running a program, you get an error message *and* the cursor flags the error's location for you.

Only $99.95

*60-Day Money-back Guarantee***

For the dealer nearest you, or to order now,
Call (800) 543-7543

Sieve (25 iterations)

	Turbo Pascal 4.0	Turbo Pascal 3.0
Size of Executable File	2224 bytes	11682 bytes
Execution speed	9.3 seconds	9.7 seconds

Sieve of Eratosthenes, run on an 8MHz IBM AT

Since the source file above is too small to indicate a difference in compilation speed we compiled our CHESS program from Turbo Gameworks to give you a true sense of how much faster 4.0 really is!

Compilation of CHESS.PAS (5469 lines)

	Turbo Pascal 4.0	Turbo Pascal 3.0
Compilation speed	12.1 seconds	35.5 seconds
Lines per minute	27,119	9,243

CHESS.PAS compiled on an 8 MHz IBM AT

*Run on an 8MHz IBM AT

**If within 60 days of purchase this product does not perform in accordance with our claims, call our customer service department, and we will arrange a refund.

All Borland products are trademarks or registered trademarks of Borland International, Inc. Copyright ©1987 Borland International, Inc. BI 1166

BORLAND

YES! I want to upgrade to Turbo Pascal 4.0 and the 4.0 Toolboxes

If you are a registered Turbo Pascal user and have not been notified of Version 4.0 by mail, please call us at (800) 543-7543. To upgrade if you have not registered your product, just send the original registration form from your manual and payment with this completed coupon to:

Turbo Pascal 4.0 Upgrade Dept., Borland International
4585 Scotts Valley Drive, Scotts Valley, CA 95066

Name _____

Ship Address _____

City _____ State _____

Zip _____ Telephone ( ) _____

This offer is limited to one upgrade per valid registered product. It is good until November 30, 1987. Not good with any other offer from Borland. Please allow 4 to 6 weeks for delivery of Toolboxes. Outside U.S. make payments by bank draft payable in U.S. dollars drawn on a U.S. bank. CODs and purchase orders will not be accepted by Borland.

†To qualify for the upgrade price you must give the serial number of the equivalent product you are upgrading.

Please check box(es)	Suggested Retail	Upgrade Price†	Serial No.
☐ Turbo Pascal 4.0 Compiler	$ 99.95	$ 39.95	_____
☐ Turbo Pascal 4.0 Developer's Library	395.00	150.00	
(Includes Turbo Pascal Tutor and all Toolboxes; must be ordered with compiler)			
☐ Turbo Pascal Tutor	$ 69.95	$ 19.95	_____
☐ Turbo Pascal Database Toolbox	99.95	29.95	_____
☐ Turbo Pascal Graphix Toolbox	99.95	29.95	_____
☐ Turbo Pascal Editor Toolbox	99.95	29.95	_____
☐ Turbo Pascal Numerical Methods Toolbox	99.95	29.95	_____
☐ Turbo Pascal Gameworks	99.95	29.95	_____

Total product amount $ _____
CA and MA residents add sales tax $ _____
Shipping and handling* $ _____
Total amount enclosed $ _____

Please specify diskette size ☐ 5¼" ☐ 3½"
For the IBM PS/2™ and the IBM® and Compaq® families of personal computers and all 100% compatibles

Payment: ☐ VISA ☐ MC ☐ Check ☐ Bank Draft

Credit card expiration date: _____/_____

Card # _____

*In US please add $5 shipping for each product ordered or $15 for the Compiler and Developer's Library. Outside US please add $10 shipping and handling for each product ordered or $25 for the Compiler and Developer's Library.

Number 3 on the Reader Service Card

PRODUCED 2006 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED